Escuela
Politécnica
Superior

Detección táctil para agarre robótico con realimentación

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Inés Fernández Sánchez
Tutor/es:
Pablo Gil Vázquez
Guillermo Oliver Peirós

Universitat d'Alacant
Universidad de Alicante

Detección táctil para agarre robótico con realimentación

Autor

Inés Fernández Sánchez

Tutor/es

Pablo Gil Vázquez

Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal

Guillermo Oliver Peirós

Instituto Universitario de Investigación Informática

Grado en Ingeniería Robótica

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2021

Preámbulo

La tecnología, la ingeniería y en concreto la robótica, siempre han sido conceptos que despertaban gran curiosidad en mí. En especial, la sensorización de los sistemas para darles la capacidad de ”aprender” ha sido la razón principal de la elección de este trabajo. Este trabajo consiste en el análisis del sensor táctil-óptico DIGIT, un sensor más económico dentro del área de la sensorización táctil. La meta de este proyecto es aportar información a la investigación sobre estos sensores, ayudando a la democratización de la robótica, mediante el uso de técnicas de aprendizaje supervisado.

Agradecimientos

Quiero agradecerles en primer lugar el apoyo y el cariño constante a mi padre y a mi madre. Por escucharme, por hacerme reír y por levantarme cada vez que me caía. Sin vosotros no hubiese llegado a donde estoy hoy. Gracias a vuestro trabajo, esfuerzo y sacrificio he llegado a ser la persona que soy hoy.

A mis abuelos, sin su trabajo no podría haber llegado hasta aquí. En especial a mi abuela Carmen, por ser la flor y la poesía en mi vida y a mi abuela Albina, que aunque no ha podido ver terminado este trabajo, creyó todos y cada uno de los días en mí, gracias por ser esa nota de lógica en mi vida. Gracias a ambas por su esfuerzo y por ser un ejemplo a seguir, sin ellas no habría sido posible tener todas las oportunidades que se me han brindado.

A mis amigas, que aun estando lejos y no poder vernos todo lo que quisiésemos, son personas increíbles con las que he aprendido muchísimo y que han construido una gran parte de mis cimientos.

A mis compañeros de carrera y en especial a los amigos que he hecho en ella. Gracias por enseñarme tanto, ser personas maravillosas y por hacer de la carrera una buenísima experiencia.

A mi compañero en estos últimos años, por haber sido un apoyo inamovible y por acompañarme todos aquellos viernes a la biblioteca.

Por último, agradecer a mi tutor Pablo Gil, por su apoyo, por hacerme participe de este proyecto y por su paciencia a la hora de resolver mis dudas. También agradecer al resto del equipo de AUROVA por acompañarme a lo largo de este trabajo, en especial a Julio Castaño, por resolver mis preguntas y ayudarme en todo lo posible.

A mi madre y a mi padre, por todo su esfuerzo y paciencia

Índice general

Índice de figuras

Índice de tablas

1 Introducción

En este primer capítulo se introducen los objetivos del proyecto y la perspectiva desde la que se abordaron, para el desarrollo de este Trabajo de Fin de Grado (TFG) el cual se enfoca en percepción táctil para agarre robótico.

1.1 Motivación y contexto

El ser humano es una criatura con un afán inagotable de aprender y conocer el entorno que le rodea. En cada época de la historia se han explotado diferentes facetas y retos para las personas. Actualmente le toca el turno a la tecnología y más concretamente a la robótica.

La robótica es un área que se encuentra en pleno desarrollo en la actualidad. Lo que se conocía como un género dentro de la ciencia ficción es ahora uno de los campos más punteros dentro de la industria y se abre camino en entornos tan novedosos como la robótica social o la medicina.

La sensorización es la base del "aprendizaje" de los robots para poder ampliar su rango de actuación y, la dotación de "tacto" dentro de la robótica es un campo relativamente novedoso que puede mejorar las capacidades de los robots en diferentes entornos mejorando sus aptitudes. Por ejemplo, la posibilidad de poder "enseñar" mediante algoritmos a una máquina y así, poder reconocer diferentes objetos para posteriormente manipularlos de la mejor manera posible es algo que realmente suena a ciencia ficción y no puede comenzar a ser más real.

Como persona detrás de este trabajo y como ser humano, mi interés por la tecnología y sobre todo por la sensorización de los robots ha crecido durante estos años, siendo aún más creciente en estos últimos meses en los que he llevado a cabo la realización de este proyecto. El poder manipular diferentes objetos y poder obtener resultados con ello es algo que solo

alimenta mi curiosidad científica e ingenieril por este campo.

1.2 Propuesta

En este trabajo de fin de grado se da a conocer el sensor DIGIT, un sensor táctil-óptico. Este sensor pretende conseguir buenos resultados de sensorización tratando de disminuir los costes de fabricación frente a otros sensores de contacto mucho más caros. Además, aprovecha el avance tecnológico de las técnicas de aprendizaje automático a partir de imágenes para tratar de ser un sensor competitivo.

A su vez, este proyecto contempla la opción de montarlo sobre una pinza, de manera que se puedan realizar pruebas más prácticas del uso del sensor.

La investigación de estos sensores se está desarrollando en diferentes universidades, creando una comunidad de experimentación alrededor de ellos. Este TFG se realiza dentro de la investigación del grupo Automática, Robótica y Visión Artificial (AUROVA) de la Universidad de Alicante, dentro del proyecto europeo Robótica móvil colaborativa de objetos deformables en aplicaciones industriales (COMMANDIA) (Programa Interreg Sudoe, 2020) que coordina y dirige mi tutor de TFG, donde se está realizando una investigación paralela cuyo objetivo es la manipulación de objetos deformables y para ello se hace uso de percepción y control táctil.

1.3 Objetivos

En este proyecto, se lleva a cabo un desarrollo integral desde las etapas de montaje de componentes para la construcción y puesta en marcha del sensor DIGIT, hasta el desarrollo de la algoritmia para poder leer e interpretar las señales que adquiere, para así poder integrarlo junto a una pinza robótica montada en un manipulador robótico, y de este modo poderlo emplear como dispositivo de un control realimentado de una pinza o mano robótica. En definitiva, la finalidad es asentar las bases que permitan, en un futuro cercano, implementar algún método de control de manipulación y/o agarre de objetos haciendo uso de sensores DIGIT.

Por lo tanto, también es objetivo de este trabajo, comprender los modos de operación y

funcionamiento de una pinza robótica de la marca Robotiq, así como programar su uso a través de Robotic Operating System (ROS). En esta pinza se montarán, como se verá a lo largo de este trabajo, los sensores DIGIT.

Y como ya se ha comentado, desarrollar algoritmos de detección táctil y llevar a cabo diferentes pruebas que permitan compararlos, para determinar puntos a favor y en contra del uso de este sensor, así como valores de eficacia y confianza cuando éste se emplea para tareas de detección de contacto o no contacto en el proceso de agarre de distintos objetos.

Con los algoritmos desarrollados, también se pretende determinar si es posible destinar un sensor de bajo coste y de fabricación no industrializada como es el DIGIT, a tareas sencillas de percepción. De este modo, pudiera ser de interés que DIGIT pudiera servir de sustituto de otros sensores muchísimos más caros de uso comercial y de fabricación estandarizada, con el consiguiente ahorro de costes y potenciando la democratización del uso de la robótica.

1.4 Estructura del documento

En este apartado se pretende explicar, para mayor entendimiento del lector, las diferentes partes que contiene este trabajo, cómo se dividen y su contenido principal. En el capítulo 2 se verá qué avances existen hoy en día en el área de los sensores táctiles, concentrándose en los diferentes tipos existentes y en los ópticos especialmente. Además, se verá en qué consiste el aprendizaje supervisado y los diferentes tipos de aprendizajes automáticos existentes. El capítulo 3 se verá la arquitectura que se utiliza en el sistema táctil, tanto la parte más física como las diferentes herramientas utilizadas en el apartado software. En el capítulo 4 se explica el objetivo concreto del proyecto, el problema a resolver junto con los métodos utilizados y la forma de evaluar el sistema. En el capítulo 5 se encuentra la experimentación final para probar el uso y rendimiento de los sensores junto con la implementación de los sistemas del sensor y de la pinza. Por último, el capítulo 6 implica una revisión del proyecto general realizado y las conclusiones de los resultados alcanzados.

2 Estado del arte

2.1 Sensores táctiles

La tecnología que se ha ido desarrollando de la mano del ser humano, a lo largo de la historia ha estado inspirada en la biología y en la naturaleza en gran parte de los casos. Las personas observan su entorno a través de sus sentidos y crean y mejoran sistemas a partir de ese conocimiento.

Uno de los cinco sentidos presentes en el cuerpo humano es el tacto. A través de éste, las personas se comunican e interaccionan con su entorno. Si se trata de pensar en realizar cualquier interacción cotidiana sin este sentido, tareas de lo más simples se vuelven algo complejas. No se podría captar la temperatura, algo que puede resultar muy peligroso, ni tampoco la reacción a la fuerza o presión que se ejerce sobre los objetos. Esta retroalimentación nos permite adaptarnos de manera casi "automática" sin tener que pensar de forma consciente sobre las propias acciones, realizando los movimientos de forma inconsciente.

Además, el tacto nos permite reconocer formas y texturas para identificar objetos (Klatzky y cols., 1985). Incluso, se puede utilizar independiente respecto a otros sentidos, aunque en ocasiones pase desapercibido. Por tanto, al utilizar varios sentidos combinados, el reconocimiento es completo y casi perfecto.

Se puede pensar, por ejemplo, en cómo agarrar un vaso de cristal y la función que tiene el tacto en esta actividad. Si no se sujeta con una mínima fuerza se resbalará hasta caerse, pero, si se realiza demasiada fuerza, podría partirse. Además, si en un momento dado se estuviese resbalando, sin tacto no se podrían notar las fuerzas de rozamiento dinámico para frenar la caída. El tacto es, por tanto, para los humanos, una herramienta extremadamente útil para el reconocimiento y manipulación de objetos.

De forma análoga, el brazo robótico tiene una función similar al de las personas, alcanzar

y manipular objetos de la mejor manera posible. Debido a la complejidad del cuerpo, la tecnología todavía no es capaz de recrear los mismos sistemas, sin embargo, estos avances generan mejoras cada día (Velasco-Sánchez y cols., 2020). De esta forma nacen los sensores táctiles, capaces de realimentar la posición de un brazo y/o una mano robótica.

Son muchas las características que se pueden querer captar mediante los sensores y dependiendo de las necesidades es conveniente usar un tipo de sensor u otro. Algunas características y estímulos medibles son fuerza/presión, textura, rigidez, contacto, deslizamiento o posición (Pedreño Molina y cols., 2000), particularidades del agarre a tener en cuenta a la hora de establecer la sujeción de un objeto.

Existe una gran variedad de sensores táctiles en desarrollo que se pueden clasificar según la tecnología que utilizan para su funcionamiento:

- Sensores capacitivos

Los sensores capacitivos son capaces de detectar materiales metálicos y no metálicos sin que haya contacto directo con el objeto a detectar. Además, pueden detectar tanto elementos sólidos, como líquidos.

El funcionamiento de estos sensores se basa en un condensador abierto con un campo eléctrico. Al acercar un objeto, este actúa como un dieléctrico, y, debido al contacto con el campo eléctrico, se modifica el equilibrio del condensador. La variación de capacidad que se produce en ese momento se mide mediante un oscilador y se compara con un umbral para determinar la distancia.

La sensibilidad de este sensor en la detección varía dependiendo del material a detectar.

El ejemplo más común en el que se utiliza este tipo de tecnología es en las pantallas táctiles de los teléfonos móviles. Además, existen aplicaciones en investigación dentro del área de la robótica como la que se puede ver en Maiolino y cols. (2013).

- Sensores resistivos

Estos sensores utilizan una serie de láminas en las que dos de ellas contienen elementos conductores formando líneas rectas. Estas placas crean entre ellas una matriz, teniendo información en sentido x e y. Al realizar presión sobre el sensor, las placas entran en contacto

y cambian la diferencia de tensión existente. Debido a la forma de matriz que se comentaba, se puede obtener el punto en el que se realiza esta fuerza.

La desventaja de este sistema es la reiterada deformación de la capa superior debido a la fuerza externa ejercida.

En Schöpfer y cols. (2010) se puede ver un ejemplo del uso de este tipo de sensores en el ámbito táctil robótico.

- Sensores piezoeléctricos

Estos sensores aprovechan las cualidades de algunos materiales, como ciertos cristales, elementos cerámicos o polímeros, para generar una carga eléctrica cuando son sometidos a estrés al ser deformados. La sensibilidad que tendrá el sensor depende de la forma en la que ha sido cortado el material a utilizar.

Estos sensores suelen tener una gran confianza y precisión, siendo sensores bastante utilizados. Además, tienen una respuesta dinámica más rápida que los sensores capacitivos. Sin embargo, estos sensores pueden tener un cierto error al aplicarse una tensión estática, siendo empleados solo para mediciones dinámicas. En general, son bastante sensibles a la temperatura y las uniones eléctricas son frágiles pudiendo producir roturas.

En Kolesar y Dyson (1995) se puede leer una investigación sobre un sensor piezoeléctrico utilizado para reconocer siluetas de ciertas figuras geométricas.

- Sensores barométricos

Estos sensores utilizan unos transductores que miden la presión, a la que se somete un líquido o aire, ejercida por un objeto.

Al someter un líquido que se encuentra en posición de reposo a una fuerza, se producen unas vibraciones con las que se puede medir la presión ejercida. Al utilizar un material líquido para esto, se obtiene una respuesta de alta frecuencia a la vez que se consigue la deformación del sensor añadiendo adaptabilidad, pudiendo medir un amplio rango de presiones junto con la dirección en la que se ejerce la fuerza.

En otros casos se utiliza simplemente un material elástico, como una goma, en el que este actúa como una membrana, que al deformarse cambia el valor de presión medido por el transductor.

Los sensores con líquido tienen una respuesta de mayor frecuencia en comparación a la versión de goma, pero, estos últimos tienen un menor coste. Por tanto, el uso de unos u otros dependerá de los requerimientos de la velocidad de respuesta que se necesitan en la aplicación deseada.

El mayor inconveniente de este tipo de sensores tan versátiles es su precio, que normalmente es bastante alto por cada dispositivo que se quiera utilizar.

Uno de los ejemplos más conocidos y que utiliza la versión líquida de estos sensores es el BioTac (Fishel y Loeb, 2012).

- Sensores ópticos

Los sensores táctiles ópticos utilizan una cámara para captar imágenes con las que clasificar las características presentes ante la situación en la que se quiere usar el sentido del tacto artificial.

En los últimos años las cámaras han tenido un alto desarrollo y un abaratamiento de sus precios. A su vez, las técnicas de análisis de imágenes se han ido perfeccionando, mejorando el procesado de estas y ayudando a generalizar el tipo de tarea que se puede realizar.

Además de estas mejoras, los sensores ópticos en general tienen mayor sensibilidad frente a los cambios y mejor resolución espacial gracias a la cantidad de píxeles que tienen las cámaras actualmente, incluso las de bajo coste.

Uno de los inconvenientes de este tipo de sensores es que la cámara debe tener cierta distancia con la superficie de contacto. Teniendo en cuenta que el sensor en muchas ocasiones debe ir incorporado en otro sistema, como la mano de un robot, se necesita que el robot tenga los dedos lo suficientemente grandes para integrarlo, por lo que es un factor importante su diseño y cuanto más pequeño sea más realista y sencillo de integrar será. Además, por esta razón, no son utilizados en tareas de precisión actualmente por el volumen que ocupan. Existen tres posibles configuraciones de sensores ópticos según Shimonomura (2019):

  - Platos conductores de luz

Consisten en una cámara, un material conductor de luz, un Light Emitting Diode (LED) y un elastómero. La luz del LED viaja a través del conductor lumínico de manera constate.

Existen dos formas, con y sin elastómero. En el caso de tenerlo, al ejercer una fuerza exterior sobre él, se produce un cambio en la imagen reflejada que capta la cámara y que es debido a la deformación de éste. En caso de no tenerlo, la luz incide sobre el objeto externo reflejando su imagen directamente y siendo captada por la cámara.

– Con desplazamiento de marcadores

En este sensor, existen unos marcadores colocados en el elastómero, que, al ser presionado hace que se desplacen de lugar. El sistema conoce la posición estándar de los marcadores y, con la presión, la cámara capta el cambio de posición de los marcadores, como consecuencia de la deformación del elastómero. Esto permite relacionar la deformación y dirección de la fuerza con el desplazamiento de los marcadores.

– Membrana reflectora

La luz de varios LEDs recorre el elastómero y, al introducir una fuerza que lo deforma, la luz se refleja gracias a un pigmento reflector, captando la variación y la forma en la que cambia al incidir sobre la cámara.

Estas categoría no son excluyentes y la mayor parte de sensores pueden trabajar con estas tecnologías mezcladas entre sí. Así por ejemplo, se puede utilizar una membrana reflectora a la que se le añadan marcadores, de manera que captará el cambio de forma y la variación de posición de los marcadores. La decisión sobre los usos dependerá de las técnicas que se utilizarán y el objetivo final que se quiera lograr.

Algunos ejemplos de sensores que utilizan un sistema táctil óptico son TacTip (Ward-Cherrier y cols., 2018) o GelSight (J. Li y cols., 2018) entre otros (Shimonomura, 2019)(Sferrazza y D'Andrea, 2019), además del propio sensor DIGIT (Lambeta y cols., 2020), objeto de estudio de este proyecto.

A parte de todos estos tipos de sensores, también existen otros que no son tan ampliamente utilizados dentro del área táctil, como son los inductivos, basados en magnetismo o en ultrasonido, pero por sus diferencias de utilidad con los empleados en este trabajo no se han abordado.

2.2 Aprendizaje supervisado

El aprendizaje automático nace a partir de la idea de conferir "inteligencia" a las máquinas. Esta industria consiste en desarrollar algoritmos que permitan a los sistemas funcionar sin ser directamente programados para realizar una tarea específica y que creen un modelo generalizable con el que poder realizar predicciones.

El aprendizaje supervisado, en el que se basa este proyecto, es una técnica incluida dentro de los sistemas de aprendizaje automático.

Los métodos de aprendizaje automático son apropiados para utilizar en aplicaciones donde no se le puede dar especificaciones concretas para el comportamiento de un programa, pero donde existe la posibilidad de ejemplificar los procesos (Learned-Miller, 2014).

Por poner un ejemplo sencillo, son técnicas como las que se utilizan para recomendar anuncios, música, vídeos… a partir del uso diario de un usuario. Estos sistemas aprenden y clasifican las preferencias de la persona para ofrecer contenido que pueda interesarle. Como se puede intuir, estos sistemas se encuentran en infinidad de aplicaciones actualmente y cobran una gran importancia debido a la capacidad de procesamiento de datos que realizan.

El aprendizaje automático se divide en diferentes ramas a su vez (Ayodele, 2010):

- Aprendizaje supervisado

En el aprendizaje supervisado, se utilizan datos previamente conocidos y etiquetados. Las muestras que se utilizan para el entrenamiento tienen la forma (x_i,y_i), donde x_i es un vector o matriz de n dimensiones, mientras que y_i es un valor escalar (Learned-Miller, 2014) etiquetado por un humano. Los datos de tipo x_i suelen dar las características con las que se va a tratar de clasificar los ejemplares conocidos, mientras que y_i explicita de qué tipo es la x_i a la que va ligada. De esta manera se explica al modelo qué características corresponden a cada modelo, para poder clasificar muestras cuyas etiquetas no se conocen.

Dentro del aprendizaje supervisado, se encuentran a su vez, dos subcategorías: algoritmos de regresión y de clasificación.

Por un lado, los modelos de regresión analizan la dependencia de una variable sobre otra variable independiente. Es decir, estudian cómo varía una de las variables en función del cambio de variables que no dependen del resto. De esta manera, se crea un modelo en el

que, al introducir una variable dependiente, se puede obtener su variable independiente, un resultado antes desconocido.

Algunos ejemplos que utilizan esta técnica son los algoritmos de regresión con aproximación lineal, árboles de decisión o máquinas de soporte vectorial.

Por otro lado, los modelos de clasificación, como su nombre indica, tratan de crear un modelo a partir de muestras conocidas que pertenecen a distintos grupos. Este modelo determinará en qué lugar se puede identificar una cierta variable desconocida.

Algunos ejemplos pueden ser algoritmos de clasificación de árboles de decisión, máquinas de soporte vectorial o K vecinos más cercanos. Posteriormente se hablará más en profundidad de algunos de estos ejemplos.

Un ejemplo de la utilización de los sistemas de aprendizaje supervisado mediante modelos de clasificación es Hu y cols. (2014), en el que se utilizan estos algoritmos, haciendo uso de la técnica de Support Vector Machine (SVM), para el reconocimiento de diferentes superficies de telas para acercar el uso del tacto humano a un robot.

Los modelos de clasificación utilizan una serie de conceptos básicos que se verán para mayor entendimiento a lo largo del documento:

  - Muestras: elementos que se quieren reconocer o que ya son conocidas. Cada dato contiene una serie de características por las que se le puede identificar.

  - Etiquetas: normalmente es un escalar con el que se define a que clase pertenece cada muestra.

  - Query: muestra que no se utiliza para entrenar el modelo y que no es conocida, por lo que se quiere definir su clase a partir del modelo creado.

- Aprendizaje no supervisado

En este tipo de algoritmos, la clasificación de la salida no es conocida y de igual manera, las etiquetas también se desconocen. En este caso, el sistema debe crear una clasificación a partir de la semejanza en las características y los patrones que estas crean de los datos introducidos. Así, los humanos no necesitan intervenir etiquetando las muestras previamente.

Se puede intuir, que la salida de los datos no será una clase en particular, si no que, al usar estos algoritmos, se obtienen agrupaciones de las entradas a partir de la interpretación que realiza el modelo.

Algunos de los algoritmos más representativos de este tipo de aprendizaje son los llamados de tipo clustering. Algunos ejemplos englobados dentro de este son los algoritmos K-means, Principal Component Analysis (PCA) o Individual Component Analysis (ICA).

En Lee y Lewicki (2002) se puede ver un ejemplo de uso de la técnica de aprendizaje no supervisado para la clasificación de imágenes.

- Aprendizaje semi-supervisado

El aprendizaje semi-supervisado consiste en utilizar la base de los algoritmos supervisados y no supervisados. Para estos casos, se utilizan datasets en los que existen muestras etiquetadas y no etiquetadas.

En ocasiones no es viable etiquetar todas las imágenes que no tienen relacionadas sus clases, pero, no utilizar esos datos, aunque no estén etiquetados, es una pérdida de información. En este tipo de situaciones, se utiliza el aprendizaje semi-supervisado.

De forma general, el proceso consiste en entrenar un modelo con las muestras ya clasificadas, como en el caso del aprendizaje supervisado. Una vez realizado este paso, se procede a predecir a qué clase pertenecen las imágenes no etiquetadas. Tras ser clasificadas, las muestras se denominan pseudo-etiquetadas. Finalmente, se entrena otro modelo con las muestras etiquetadas y pseudo-etiquetadas que se utilizará como modelo concluyente.

En conclusión, estos algoritmos tratan de potenciar el uso de un conjunto más grande de muestras no etiquetadas con la ayuda de un menor porcentaje de muestras etiquetadas, tratando de mejorar los resultados que se obtendrían al usar ambos datasets por separado.

En Mallapragada y cols. (2008) se puede observar un trabajo de investigación relacionado con optimizar y obtener el mayor rendimiento a partir de casos con este tipo de datasets.

- Aprendizaje por refuerzo

Este tipo de algoritmos se utiliza en muchas ocasiones en el diseño de comportamientos que, mediante programación tradicional, serían más complejos de realizar. Para evitar esta línea de programación, el modelo debe aprender de la propia experiencia al realizar una actividad.

Para realizar este aprendizaje el sistema debe realizar una acción para pasar del estado actual a un estado siguiente. Al realizar esta acción, el agente obtiene una recompensa. Esta puede ser positiva, si se quiere reforzar esa acción o negativa, si es una acción que se quiere castigar. La meta final del algoritmo es obtener la mayor recompensa posible.

Para decidir qué acción ejecutar en cada estado debe definirse una política, que es la estrategia que debe seguir el agente para conseguir la recompensa final. Normalmente, esta política óptima es desconocida, cuando el entorno no es conocido. Para obtenerla, se deben utilizar dos pasos: exploración y explotación. Durante la exploración se realizan diferentes combinaciones con distintas recompensas, mientras que, en la explotación, se reiteran aquellas acciones que obtienen un mejor resultado. Estos pasos deben tener un equilibrio para encontrar la mejor política y obtener el número más alto de recompensas posibles.

Dos de los algoritmos base más conocidos del aprendizaje por refuerzo son Monte Carlo y Q-Learning.

En el artículo de Kober y cols. (2013) se expone la relación entre el aprendizaje por refuerzo y la robótica, incluyendo un ejemplo en el que se pone a prueba el uso de estos dos campos combinados.

Además de estos tipos de algoritmos de aprendizaje automático existen otros menos utilizados como son las técnicas de “transduction” y “learning to learn”.

En los siguientes apartados se verán algunos algoritmos, que se han utilizado en el proyecto, más en profundidad. Todos estos, se clasifican dentro de los algoritmos de aprendizaje supervisado.

2.2.1 K-nearest neighbours

K Nearest Neighbours (KNN), es un algoritmo de aprendizaje supervisado que se encuadra dentro de los sistemas de clasificación. En este algoritmo, se introducen diferentes vectores que van asociados a diferentes clases. El número de clases será, por tanto, el número de salidas posibles. Con estos datos se realiza el entrenamiento del modelo.

Al introducir un vector desconocido se utilizará el modelo para interpretar a qué clase corresponde. Esto se realiza observando un número k de los vecinos más cercanos, siendo estos

vecinos, muestras con clases conocidas dentro del modelo de entrenamiento creado. Como se puede ver en la Figura 2.1 de ejemplo, si se introdujese k=1, la muestra se clasificaría como la clase triángulo, en caso de que k fuese 2, se produciría un empate sobre la clase cuadrado y triángulo por lo que habría que introducir un método para designar una u otra. En el tercer caso k=3, se deshace el empate clasificando la muestra como cuadrado.

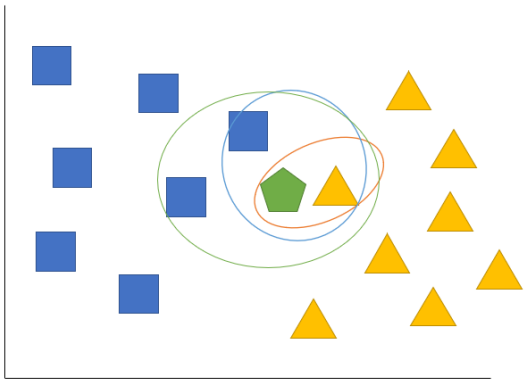


Figura 2.1: Representación del número de vecinos. Círculos: naranja: k=1, azul: k=2, verde: k=3.

La clasificación como se ha visto depende del valor de k, que suele ser un número impar para evitar casos de empate.

El valor de k no es fijo y normalmente se debe buscar en base a pruebas hasta obtener el mejor resultado posible. Sin embargo, en líneas generales, al establecer valores muy bajos de k pueden tener ruido que afecte a la clasificación, y por otro lado, los valores muy altos, en casos en los que alguna categoría tenga pocas muestras, puede no ser seleccionada al verse sobrecogida por otras clases con más datos.

En síntesis, este algoritmo es un modelo muy simple pero que llega a dar resultados altamente competitivos. En la teoría, el error asintótico al aumentar el número de muestras de entrada se acerca al error óptimo de Bayes y tiende a él cuando el número de k aumenta. Por esta razón, este algoritmo se ha convertido en un sistema de comparación para nuevos clasificadores, como las redes neuronales (Laaksonen y Oja, 1996).

El problema principal del algoritmo es que utilizará el modelo cada vez que haga una predicción y, al tener un número alto de muestras, se traduce en una elevada complejidad computacional.

2.2.2 Random Forest

El algoritmo de Random Forest (RF) se encuentra clasificado dentro de los métodos de aprendizaje supervisado. Éste, utiliza el sistema implementado en los algoritmos de árboles de decisión, donde se generan, a partir de unos datos de entrada, una serie de condiciones para clasificar estas muestras. Al final de cada árbol, se obtiene una clase a la que pertenece el dato introducido como entrada.

Los Random Forest, utilizan como base este sistema, pero en vez de utilizar un solo árbol, utilizan un número n de ellos. Para cada árbol generado se observa una condición concreta escogida de forma aleatoria. Una vez generadas las salidas se cuentan el número de veces que ha sido seleccionada cada clase y se escoge la que más votos haya obtenido. Por ejemplo, se puede observar la Figura 2.2 en la que dos de los tres árboles deciden que la salida sería de la clase naranja mientras que el restante obtiene la clase verde. Por las reglas de este método, la salida final sería la clase naranja.

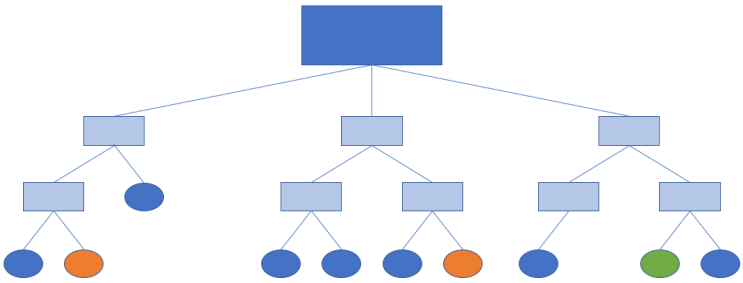


Figura 2.2: Representación esquemática del algoritmo Random forest.

A su vez, estos sistemas de clasificación han tenido sus propias variantes utilizando diferentes métodos de aleatoriedad en las condiciones (Breiman, 2001).

Esta técnica, a diferencia de los árboles de decisión, es más complicada de comprender e interpretar, sin embargo, obtiene por normal general muy buenos resultados incluso con grandes cantidades de datos.

2.2.3 Support Vector Machine

Al igual que las anteriores técnicas, SVM también se enmarca en el aprendizaje supervisado.

Este tipo de algoritmo funciona encontrando la función que separe las clases de los datos introducidos en el entrenamiento. De esta manera, al introducir una entrada no conocida, se podrá clasificar respecto de la función hallada.

Para realizar esta separación, la función que se busca se denomina hiperplano. En caso de separar los datos en un sistema de dos dimensiones, el hiperplano será una recta, mientras que, si se encuentra en tres dimensiones, la separación vendrá dada por un plano.

Si se considera un ejemplo como el de la Figura 2.3, con un hiperplano en forma de recta, se puede observar que puede haber muchas formas de separar las muestras, sin embargo, esta técnica busca maximizar la distancia entre las muestras. Para esto, se utilizan los vectores de soporte, las funciones paralelas al hiperplano que se encuentran en el margen de cada clase. Así, se busca que el margen entre los vectores de soporte sea máximo.

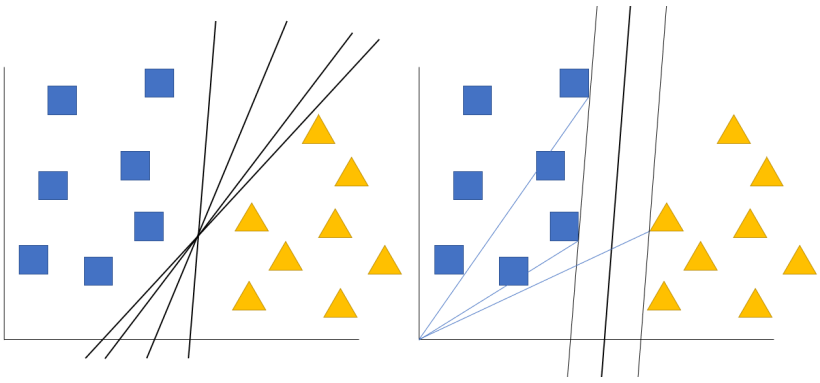


Figura 2.3: Representación del algoritmo Support Vector Machine. A la izquierda, varias posibilidades de hiperplanos. A la derecha, hiperplano con margen máximo.

En el caso de que los datos no se puedan separar de manera lineal, se utiliza la función kernel, que encuentra una solución al problema añadiendo otra dimensión al problema de manera que se puedan dividir los datos, como se puede ver en la Figura 2.4.

El algoritmo SVM obtiene muy buenos resultados con casos en los que los datos son linealmente separables. A su vez, las resultas de esta técnica son buenas en casos con varias dimensiones espaciales incluso aunque sea mayores que el número de muestras. Sin embargo, los resultados no son tan buenos en casos en los que el número de muestras es menor que el

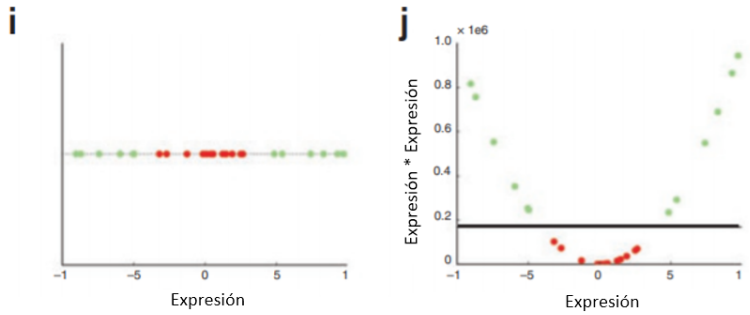


Figura 2.4: Representación teniendo en cuenta otra dimensión. Fragmento de imagen cortesía de Noble (2006).

número de clases y tampoco se obtienen directamente las probabilidades, que se tienen que obtener con un cálculo de validación cruzada muy costoso computacionalmente.

En Kim y cols. (2002) y en S. Li y cols. (2003) se pueden ver ejemplos de su uso para la clasificación de diferentes texturas.

3 Arquitectura táctil

En esta sección se procederá a explicar las diferentes partes que componen el proyecto táctil. Principalmente se puede dividir en la parte hardware, compuesta por las partes físicas como el sensor o la pinza y la parte software, donde se procederá a explicar los sistemas que hacen funcionar los dispositivos.

3.1 Hardware

En este apartado se va a ver la parte material del sistema pensado para la utilización del sensor. Notar que, parte del proyecto ha consistido en la construcción de algunas partes del sensor, tales como carcasa y elastómero, así como el ensamblaje de todas las partes y la puesta en funcionamiento del sistema físico táctil.

3.1.1 Pinza

Para el agarre de los distintos objetos se utiliza una pinza de la marca Robotiq (*Robotiq*, 2020).

Robotiq es una empresa dedicada a la fabricación y venta de efectores finales en su mayor parte para utilizar en diferentes brazos robóticos.

El modelo utilizado para este proyecto es la pinza 2-Finger 140. Su nombre se debe a la longitud en milímetros que existe entre sus dedos cuando la pinza está completamente abierta. Se puede ver un esquema representativo de la pinza abierta en la Figura 3.1.

Este modelo cuenta con dos dedos articulados que tienen, cada uno, dos articulaciones. Los dedos son subactuados, debido a que se tienen menos actuadores, en este caso un motor, que grados de libertad.

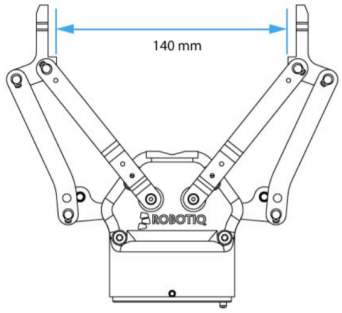


Figura 3.1: Esquema de la pinza 2-Finger 140 en posición abierta.Imagen cortesía de *Robotiq* (2020).

A su vez, como se ha mencionado antes la distancia entre dedos al estar abierta es de 140mm. Abierta cuenta con unos 209.4 mm de alto y cerrada con 232.8 mm. El resto de las medidas pueden verse en la Figura 3.2.

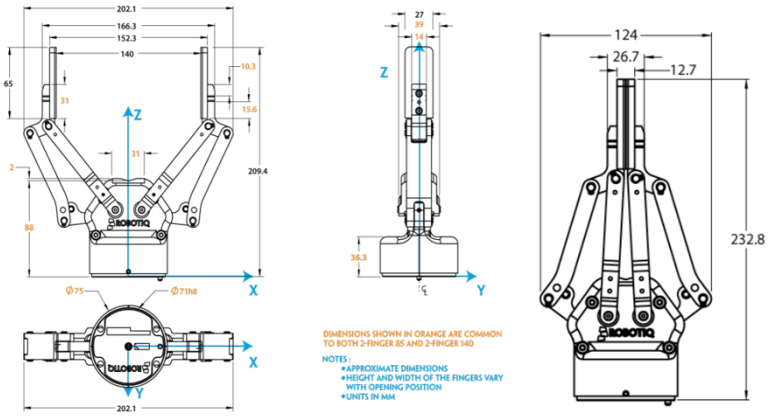


Figura 3.2: Cotas de la pinza 2-Finger 140. Imagen cortesía de *Robotiq* (2020).

Dependiendo del objeto que se necesite aferrar, la pinza utiliza un sistema de cierre adaptativo automático de manera que puede capturar los objetos cerrándose de manera paralela a ellos o entorno a ellos.

A su vez se puede utilizar el extremo de la pinza para coger los diferentes objetos de manera externa a ellos, el uso más estándar, o de manera interna si la geometría de los objetos lo permite. Para este proyecto se ha utilizado un cierre interno para hacer el agarre ya que los sensores se colocaron en las caras internas de los dedos de la pinza.

La ejemplificación de estos cierres puede verse en la Figura 3.3. En esta imagen se pueden

ver los movimiento con la pinza 2-finger 85, sin embargo, se pueden realizar de la misma manera con el modelo 2-finger 140.

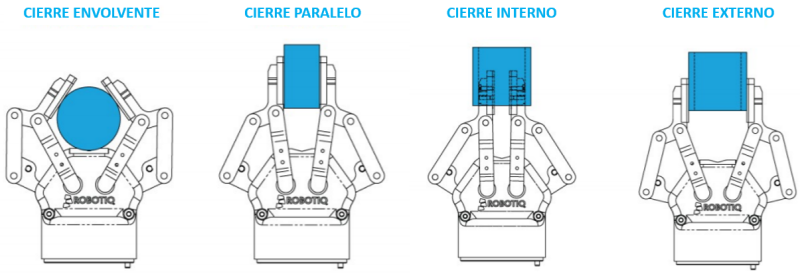


Figura 3.3: Ejemplos de cierres de la pinza. Imagen cortesía de *Robotiq* (2020).

Toda la información respecto a la pinza puede encontrarse en el documento informativo de este modelo y su paralelo 2-finger 85 (*Robotiq*, 2020).

3.1.2 Sensor

DIGIT es un sensor táctil óptico, debido a que utiliza imágenes para el reconocimiento del comportamiento de los objetos en el agarre. Gracias a las imágenes captadas se pueden clasificar las características que presentan creando un modelo para la tarea.

Este sensor consiste en una estructura compuesta por un soporte físico creado con una impresora 3D, una cámara tipo Printed Circuit Board (PCB), una tira con tres LEDs, una placa de metacrilato transparente y un elastómero creado manualmente.

El diseño de la carcasa tendrá, una vez montada, unas dimensiones de 20x27x18 mm siendo estas medidas ancho, altura y profundidad respectivamente. Estas medidas son relativamente pequeñas, aumentando la capacidad del sensor de ser incorporados en sistemas tales como manos robóticas, siendo el tamaño de los sensores, un problema que se trata de solventar actualmente en el área de la robótica. A su vez, la carcasa consta de tres partes que se explicarán en mayor profundidad posteriormente. Al ser modelos impresos en 3D, se pueden realizar modificaciones de la parte intermedia aumentando o disminuyendo su longitud, ajustándola a las necesidades personales del proyecto. Sin embargo, estos cambios deben tenerse en cuenta debido a que afectan a la distancia focal entre la cámara y la zona de agarre.

La cámara integrada permite un máximo de hasta 60 fps, aunque para este proyecto se

utilizó una velocidad de 30 fps, para captar los frames de vídeo. Estas imágenes tienen una resolución de 240x320 píxeles. Los LEDs que incorpora el sensor tienen la misión de iluminar correctamente la zona interior en la que se verá la forma y efecto que crea el objeto sostenido sobre el elastómero.

Debe tenerse en cuenta que para conectar el sensor a un ordenador debe utilizarse un cable y entrada USB 3.0.

Como se ha mencionado antes el sensor consta de diferentes partes. Algunas de ellas se han debido crear o montar de manera manual en el proyecto, por lo que a continuación se realizará un seguimiento en profundidad de ellas.

3.1.2.1 Montaje del sensor

La parte exterior puede construirse con una impresora 3D como sería en este caso. Para ello pueden consultarse los archivos proporcionados por DIGIT (Lambeta y cols., 2020).

La carcasa exterior se divide en tres partes (Figura 3.4). La parte posterior, donde se aloja la placa, la parte intermedia, donde se sujetan los LEDs y la delantera donde se coloca placa de acrílico junto con el gel.

Figura 3.4: Secciones de la carcasa DIGIT.

Durante el proceso de montaje se probaron diferentes materiales para las carcasas. El primer prototipo se construyó con plástico Polylactic Acid (PLA) (ácido poliláctico). Este material era bueno para ajustar las medidas finales ya que era muy fácil de limar, sin embargo, el acabado era bastante imperfecto y se veía poco estética. A su vez, el color de esta carcasa era de color amarillo, lo que tampoco era conveniente a la hora de captar las imágenes por ser un material demasiado claro.

El segundo material que se utilizó fue una resina de color grisáceo. Este material tenía un acabado mucho más vistoso, pero traía ciertas desventajas. Al construir las piezas con la resina, esta se expandía al secarse, haciendo que las medidas fueran bastante inexactas, algo bastante importante teniendo en cuenta las medidas del sensor. Esto producía que tuviese que eliminarse una parte importante de la carcasa siendo además este material muy complicado de limar. Por último, otro contra de este material era su fragilidad, que, mientras se limaba se pudo comprobar cómo se rompían en ocasiones ciertas zonas.

Por último, se probó a imprimirse en Acrylonitrile Butadiene Styrene (ABS) (acrilonitrato butadieno estireno). Esta última prueba terminó siendo un acierto ya que el material tenía un acabado muy bueno, era lo suficientemente resistente y apenas fue necesario hacer ajustes para introducir la placa en su interior.

El tipo de carcasa es importante debido a que esta puede influenciar la imagen que capta el sensor. El material y color puede afectar a la transmisión de luz dependiendo de la opacidad que aporta al interior del sensor.

Respecto a la placa DIGIT, esta contiene la circuitería necesaria para hacer funcionar el sensor. Tiene tres partes principales a tener en cuenta a la hora del montaje. Por un lado, la entrada para conectar el cable al ordenador, un conector para los LEDs y una cámara que debe quedar frontalmente para conseguir información del sensor. La placa y sus partes pueden observarse en la Figura 3.5.

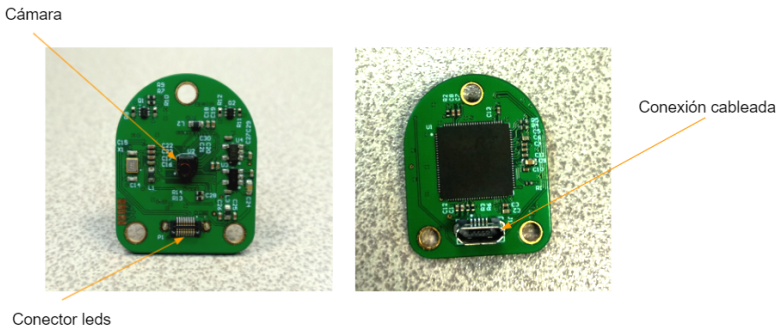


Figura 3.5: Partes de la placa DIGIT.

Una vez se tiene la carcasa impresa, se coloca la placa DIGIT en el interior, de manera que la entrada para la conexión mediante cable salga por la parte posterior y la cámara

quede hacia delante. Se debe colocar con cuidado debido a que la circuitería es delicada. A continuación, se conecta la tira de LEDs a la placa y se introduce la parte intermedia de la carcasa, haciéndolos pasar a través de ella. Los LEDs deben colocarse entre la carcasa y unas pestañas que sirven para sujetarlos sin que se muevan tal y como se puede ver en la Figura 3.6.

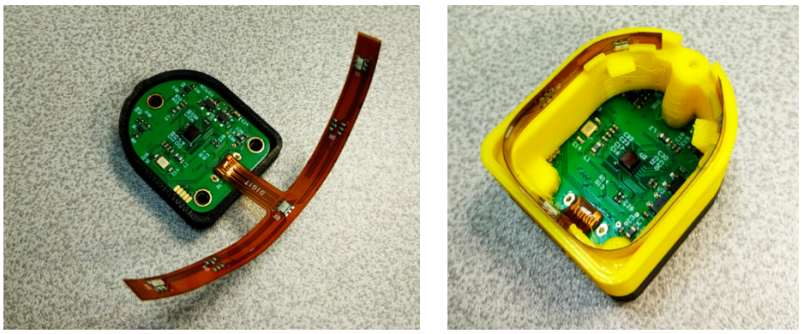


Figura 3.6: Imagen de los LEDs y su montaje en el sensor.

Una vez situados los LEDs, se coloca la última parte de la carcasa. Dentro de esta, debe posicionarse la plancha de metacrilato cortada. Esta plancha, totalmente transparente, sirve para situar el gel encima de ella y evitar que entre polvo al interior.

Una vez realizados los diferentes pasos, solo faltaría el gel para completar el montaje del sensor, quedando el resultado final como se puede ver en la Figura 3.7.

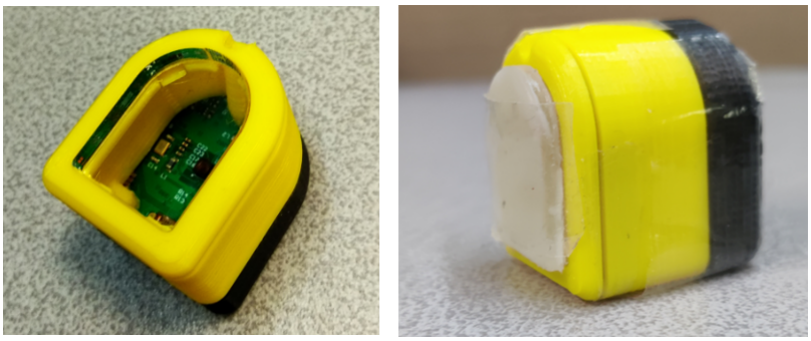


Figura 3.7: Imagen del sensor montado sin gel (izquierda) y con gel (derecha).

3.1.2.2 Creación del gel

El gel que se emplea en el sensor DIGIT, se coloca en la parte frontal de este, en la zona de contacto con los objetos a sujetar. Su tarea consiste en mejorar el agarre de las piezas haciendo que no se resbalen al sujetarlas. Además, permite la visualización de las imágenes por la cámara del sensor. Si el gel fuese demasiado opaco, la cámara no podría grabar imágenes, pero, si fuese demasiado transparente, no se podrían ver las marcas que deja el agarre y por tanto, no se podría saber si habría contacto o no. Por todo esto, el gel es una parte realmente importante del sensor.

Para crear el gel se necesitan una serie de productos que se pueden ver en la Figura 3.8.

(a) **(b)**

Figura 3.8: (a) Materiales para la fabricación de la pintura blanca. (b) Materiales para la creación del gel base junto con el Slacker.

La preparación del gel cuenta con dos partes: la mezcla para la pintura blanca y para el gel en sí.

Pintura blanca

La pintura blanca sirve para que el gel no quede totalmente transparente haciendo que las imágenes se vean menos nítidas pero produciendo que la presión en el gel se haga más fácil de leer al entrar en contacto con un objeto.

Primeramente, se miden dos partes iguales de EcoFlex 00-10 A y B para después mezclarse.

A continuación, se mide la cantidad de pigmento blanco en proporción a lo utilizado de EcoFlex. Para ello el pigmento blanco debe ser un 3% de la mezcla total anterior. Debido a la proporción tan pequeña de pigmento que se utiliza, se puede ver la cantidad aproximada

que se utilizó en la Figura 3.9.

Figura 3.9: Imagen de la cantidad de pigmento blanco aportada a la mezcla.

Una vez mezclados estos tres componentes se utiliza un 20% del peso de ellos de NOVACS. Este producto permite reducir la densidad haciendo la mezcla más ligera. Se puede añadir lentamente para observar cómo cambia la textura.

Una vez mezcladas estos cuatro componentes ya se tiene la pintura blanca lista.

Fabricación del gel

Para fabricar el gel se necesita un molde con el que darle la forma final deseada. En este caso se utilizó un molde mecanizado de aluminio (Figura 3.10).

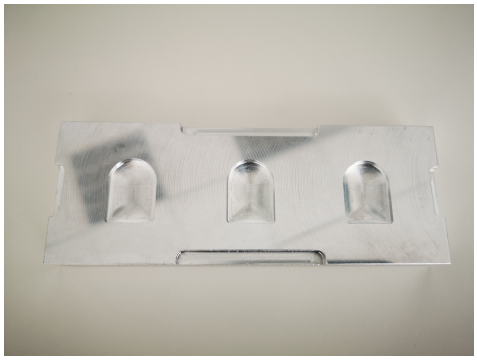


Figura 3.10: Imagen del molde utilizado para dar forma a los geles.

El molde debe ser limpiado con alcohol antes de colocar las mezclas para evitar que haya ningún tipo de suciedad o de polvo que pueda luego quedarse en las láminas de gel y que dañen las imágenes que tome el sensor.

Una vez limpio, se debe verter una mezcla de Solaris A y B a partes iguales. Al dejarlo en

el molde se debe limpiar el material que sobresalga de el para que luego no tenga defectos y se deja secar unas 24 horas.

Una variación interesante de la mezcla de Solaris puede ser añadir Slacker. Este producto aumenta la densidad final del gel al combinarse con el Solaris. Este cambio puede ser interesante dependiendo del tipo de agarre que se quiera realizar con el sensor. En este caso, se mezcló un 90% de Solaris (45% de parte A y 45% de la parte B) con un 10% de Slacker. Con este porcentaje se obtuvo un cambio bastante sutil en la densidad por lo que si se quiere mayor rigidez debe aumentarse la cantidad de Slacker.

Al cabo de este tiempo, la mezcla debe estar seca y se puede sacar del molde. En esta parte se debe tener bastante cuidado ya que puede suceder que se cree vacío y se pueda romper si se saca demasiado rápido.

Una vez fuera, se aplica una capa de Inhibit-X sobre las yemas con un pincel y se deja secar durante media hora. Esta espera es un buen momento para preparar la pintura blanca explicada anteriormente.

En este caso, como se ha visto, se utilizó un pincel para aplicar el producto, pero, tanto en este paso como en el siguiente lo más indicado habría sido usar un aerógrafo para un resultado más profesional.

Transcurrido el tiempo de secado, se pinta una capa de pintura sobre las yemas de los dedos. En caso de haber usado el aerógrafo, se puede dar una capa, volver a aplicar Inhibit-X y aplicar otra capa de pintura blanca.

Para secar la pintura se puede ayudar de una fuente de calor como puede ser un secador.

Una vez obtenido el gel, se puede adherir a la placa transparente con una silicona especial para no dejar marcas en la imagen. Se coloca una gota y se aprieta de manera que no quede silicona amontonada y termine quedando transparente.

Tras haber seguido todos los pasos correspondientes ya se pudo colocar el gel sobre el sensor para poder utilizarlo.

3.1.3 Sensor montado en pinza

Debido a la necesidad de montar los sensores sobre los dedos de la pinza para poder probar el sistema de agarre dotándolo de sensorización táctil, se ha requerido la construcción de un

adaptador de tipo dedal que permitiese anclar cada sensor táctil a cada dedo, sin restarle capacidades cinemáticas (movilidad de apertura y cierre) y dinámicas (aumento de peso de carga) a la mecánica de la pinza. Este adaptador se diseñó a través de un modelo en formato Computer-Aided Design (CAD) y se construyó mediante impresión 3D.

El modelo CAD está inspirado en el modelo propuesto por Facebook para los DIGIT. El modelo que se facilitaba con los sensores no permitía su correcto anclaje en la versión de la pinza ROBOTIQ de la que dispone el grupo AUROVA de la Universidad de Alicante, lo que ha obligado a modificar el diseño para que pudiera ser montado. El diseño se ha llevado a cabo en colaboración con el grupo MACCS de SIGMA/Clermont (Francia) para que pudiera adaptarse a las dos pinzas que se dispone en ambos centros, Alicante y Clermont. En la Figura 3.11 se puede ver el modelo CAD que se utilizó en la Universidad Alicante y que se usó para sujetar el sensor a la pinza en las pruebas.

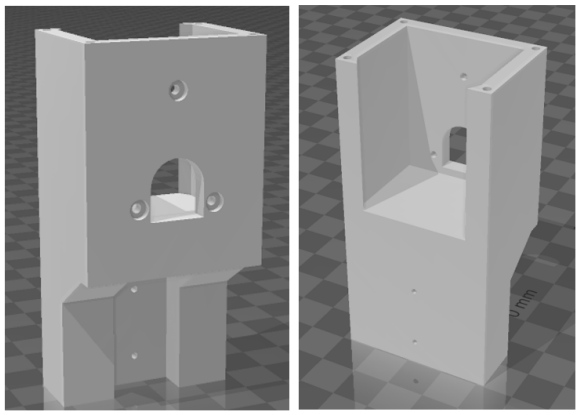


Figura 3.11: Modelo 3D del adaptador utilizado para el sensor.

3.2 Frameworks de trabajo

En este apartado se verán los diferentes entornos y herramientas de software utilizados para la realización del proyecto.

3.2.1 Ubuntu

El sistema operativo en el que se ha trabajado, tanto para implementar las herramientas de la pinza como del sensor, ha sido Ubuntu en su versión 16.04 LTS.

Se decidió utilizar este sistema debido al uso de ROS Kinetic, que se definirá en los siguientes apartados.

3.2.2 Oracle VM VirtualBox

Para implementar el sistema operativo de Ubuntu fuera del laboratorio se utilizó la máquina virtual Oracle VM VirtualBox con versión 6.1.14 (*Oracle VM VirtualBox*, 2020).

Una máquina virtual es una herramienta que permite disponer de un sistema operativo diferente dentro de un equipo con un sistema operativo huésped, que le proporciona los recursos. Esta decisión se basa en la capacidad de dotar al sistema mayor flexibilidad de trabajo ya que utilizar una máquina virtual permitía trabajar tanto dentro como fuera del laboratorio, sin instalaciones ni modificaciones de software, pudiendo llevar todo lo necesario encapsulado dentro de la máquina. Para las pruebas finales en el laboratorio se empleó una versión de Ubuntu sobre una partición del disco duro del PC que se utilizaba para controlar la pinza física real.

3.2.3 ROS Kinetic

La herramienta software que se utilizó para visualizar la pinza y poder moverla tanto física como virtualmente fue ROS Kinetic.

ROS (Robot Operating System) (*ROS*, 2020) es un entorno para la escritura de software en el área de la robótica. Su objetivo inicial era crear un sistema con el que unificar la programación de la robótica que sirviera de entorno base para comunicar procesos, tales como la parte de planificación y control de la pinza, como la parte de sensorizado y percepción

táctil.

ROS Kinetic, a su vez es una versión específica para utilizar con versiones de Ubuntu 16.

3.2.4 Anaconda

Anaconda (*Anaconda*, 2020) es un sistema que se utiliza para la gestión de paquetes. Permite la creación de diferentes entornos en los que instalar diferentes softwares evitando que puedan entrar en conflicto unos con otros.

Dentro de este proyecto toma un papel importante, ya que se utiliza dentro de la máquina virtual para utilizar la versión de Python, requerida por el sensor DIGIT para instalar los requerimientos software necesarios, firmware de DIGIT, así como, otras herramientas de trabajo como OpenCV y Scikit-Learn, sin que las versiones de librería empleadas entren en conflicto con otras versiones por defecto que pudieran estar instaladas sobre el sistema operativo Ubuntu o como parte de otros proyectos de investigación.

Se debe tener en cuenta que, en ocasiones como en este caso, dentro del entorno de Anaconda, Python puede tener algunos problemas al trabajar a la vez con ROS. Esto se puede solucionar eliminando momentáneamente algunos paquetes de ROS de la ruta del sistema, implementando las librerías necesarias y pudiendo volver a activar los paquetes de ROS posteriormente si así se desea.

3.2.5 OpenCV

Para el tratamiento de las imágenes obtenidas a partir del sensor se utiliza OpenCV (*OpenCV*, 2021), una biblioteca especializada en procesamiento de imagen y visión artificial programada en C++ pero que se puede utilizar en otros lenguajes como es el caso de Python.

Esta biblioteca se instaló dentro del entorno de Anaconda para poder manejar las imágenes de los datasets. Además, se utilizó como herramienta para el aprendizaje automático pudiéndose comparar los resultados obtenidos con otras librerías de uso empleadas como Scikit Learn. No obstante, hacer notar que OpenCV no es un software especializado en aprendizaje automático, algo que se puede observar en la implementación de los algoritmos de aprendizaje supervisado que se han desarrollado, como se verá posteriormente.

La versión utilizada en este proyecto es OpenCV, para Python, 4.5.1.48.

3.2.6 Scikit-Learn

Al igual que OpenCV, Scikit-Learn (*Scikit-Learn*, 2021) es una librería, pero, en este caso, especializada en el aprendizaje automático.

Con esta librería software se pudieron implementar los algoritmos de aprendizaje supervisado que forman parte de los métodos que se han desarrollado como parte de este proyecto, estos son KNN (K Nearest Neighbours), RF (Random Forest) y SVM (Support Vector Machine).

De igual forma que antes, se instaló en el entorno de Anaconda con la versión 0.23.2.

4 Método propuesto para detección táctil

El sensor DIGIT, como se ha podido ver anteriormente, utiliza la cámara que incorpora en su interior para captar imágenes con las que poder trabajar. En esta sección se presentarán esquemas generales de los métodos implementados en este TFG, así como se describirán tanto herramientas software como algunas técnicas usadas en el análisis, procesamiento e interpretación de las imágenes táctiles que posibilitan determinar el estado del sensor que es uno de los objetivos del presente proyecto.

El uso de un sensor táctil-visual como el que nos ocupa, posibilita implementar mecanismos de mejora para tareas de agarre cuando se hace uso de pinzas o manos robóticas y se monta este tipo de sensores en los dedos de éstas. En concreto, en este trabajo se emplearán sensores táctiles-visuales como parte de la implementación de un método propio para la detección de contacto o no, en operaciones de agarre y/o interacción de tacto con objetos cotidianos.

4.1 Objetivo deseado

En la actualidad existen y están en desarrollo diversos tipos de sensores táctiles. Existen modelos con buena funcionalidad, pero la mayor parte de ellos son muy costosos económicamente. Por ejemplo, este es el caso del famoso sensor BioTac. Por esta razón, se están investigando diferentes vías para crear sensores más accesibles, con la mayor confianza posible en sus resultados. De esta motivación nace la idea de DIGIT, la base de este proyecto, siendo un sensor con bajo precio. DIGIT fue diseñado inicialmente por la empresa Facebook. Se trata de un sensor no comercializado, cuyos productos para su fabricación han sido adquiridos por el grupo AUROVA, gracias a los contactos de investigación de mi tutor de proyecto.

El objetivo principal de este proyecto consiste en la fabricación de algunos de sus com-

ponentes, el ensamblado y montaje de otros, así como la integración con la pinza Robotiq que formará parte de la experimentación. También es un objetivo principal mostrar las herramientas software necesarias para ponerlo en funcionamiento y proporcionar soporte para el desarrollo de los métodos propuestos, en su mayoría basados en técnicas de aprendizaje automático. Para llevar esto a cabo, se han utilizado algoritmos de aprendizaje supervisado vistos en el estado del arte y se ha hecho uso de librerías tales como la propia desarrollada por los fabricantes del sensor, OpenCV y Scikit Learn. Los algoritmos empleados como base de los métodos propuestos han sido descritos y comentados en capítulos anteriores.

En posteriores apartados se hará una breve introducción a herramientas como Scikit Learn y OpenCV, librerías software necesarias para poder desarrollar técnicas de detección y realizar experimentos empleando el método científico. Las métricas de evaluación y otras funcionalidades que proporcionan han permitido observar y comparar los resultados obtenidos en la interpretación de los datos del sensor. Así, se han hecho uso de las funcionalidades de estas librerías para las predicciones, haciendo uso de conocidas métricas de evaluación para analizar los resultados obtenidos. Éstas se abordarán más adelante.

4.2 Método propuesto en Scikit-Learn

Scikit-Learn es una librería de código abierto especializada en el aprendizaje automático que se puede utilizar, al programar, en lenguaje Python.

Al estar orientada a aprendizaje automático, resulta una herramienta realmente útil para trabajar con los algoritmos vistos y que simplifica mucho diversas operaciones necesarias en la mayoría de los códigos dedicados a este tipo de trabajo. En comparación con la utilización de OpenCV, Scikit simplifica el código y aumenta el número de herramientas intrínsecas que se pueden utilizar.

En un primer momento, debe localizarse la carpeta en la que se encontrarán las imágenes que quieren utilizarse para el dataset que, posteriormente, servirán para el entrenamiento del modelo que permitirá el reconocimiento del estado del sensor.

A continuación, se cargan las imágenes con ayuda de uno o dos bucles con los que recorrer las carpetas en las que se organizan las imágenes tomadas del sensor.

Posteriormente estas imágenes se pasan a escala de grises debido a que el tratamiento de

estas en un canal es más sencillo de procesar, obtiene buenos resultados.

Sumado a esta transformación, se escalan las imágenes desde un tamaño de 240 x 320 píxeles de la imagen original a una nueva resolución de 60 x 80, tratando de mantener el ratio de escalado de la imagen original. Esta transformación es recomendable debido a que se mejorarán los tiempos de ejecución del algoritmo para la creación del modelo.

Debido a que las imágenes se obtienen en un formato matricial, se deben pasar a formato vector de modo que si la matriz tenía un tamaño 60 x 80 se pasará a un vector de tamaño 1 x 4800, como en la Figura 4.1. En resumen, cada imagen será un vector.

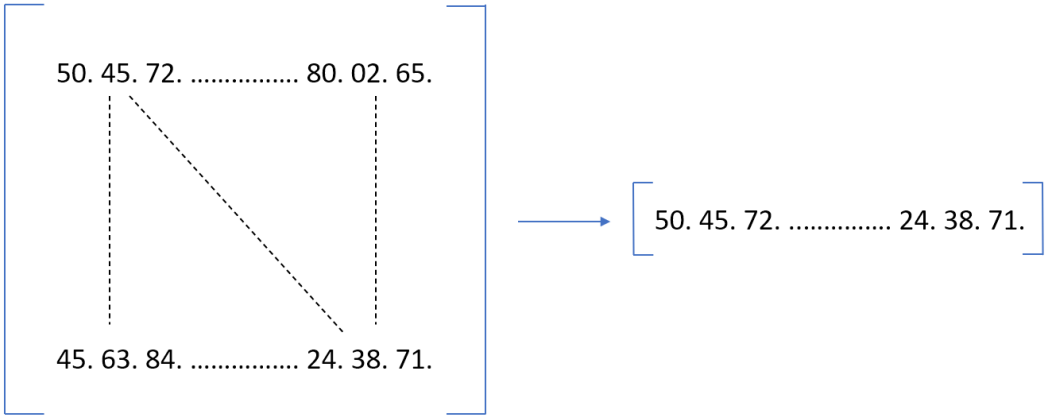


Figura 4.1: Representación del paso de la matriz imagen a formato vector.

Como el dataset necesita un gran número de imágenes, estas se irán apilando en una matriz en la que cada fila será un frame. Esta matriz se crea a partir de la librería Numpy de Python, con la que inicialmente se inicializa a ceros al ser conocido el número de imágenes que se utilizarán y la anchura de cada una de ellas, 4800, como se dijo anteriormente. Además, es necesario especificar el tipo de dato, en este caso, float32, obligatorio para que después funcione correctamente el entrenamiento de los algoritmos utilizados.

De esta misma manera, debe crearse un vector de etiquetas en el que se guarda el valor 1. o 0., dependiendo de si cada imagen muestra el sensor en posición de contacto o no, respectivamente. Esto es así, porque los métodos desarrollados se basan en aprendizaje automático supervisado. Por lo tanto, cada fila de este vector debe corresponderse con la misma fila análoga en la matriz, de manera que cada imagen siempre lleve ligada su posición de contacto o no contacto. Se puede ver una muestra impresa de la matriz de imágenes y el vector de

etiquetas en la Figura 4.2.

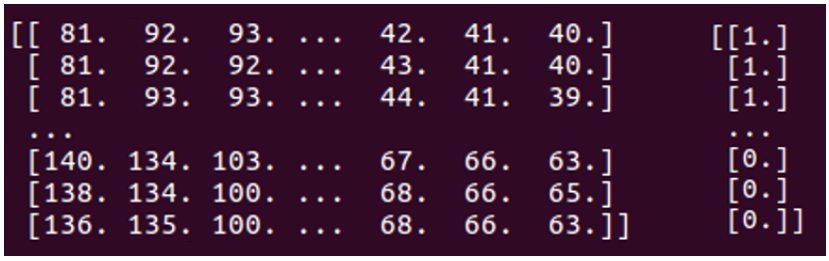


Figura 4.2: Muestra de salida por terminal de la matriz de imágenes y el vector de etiquetas correspondiente.

A partir de este punto, dependiendo del experimento a realizar existen diferencias entre los códigos empleados, sin embargo, todos siguen una línea general similar.

Primero, se debe escoger el algoritmo de aprendizaje supervisado que se quiere utilizar como base de la implementación del método propuesto. Cada uno de los algoritmos escogidos, es testeado para ver cómo se desenvuelve en la realización de tareas de clasificación. Por lo tanto, como se analizan y testean varios algoritmos candidatos, es necesario indicar los parámetros que se quieran explicitar para cada uno de ellos. Este paso se puede realizar con las propias funciones de la librería Scikit que implementa estas técnicas.

Antes de realizar el entrenamiento para la fase de aprendizaje supervisado, se utiliza una función que permite barajar las imágenes entre sí para que, al usar una parte para entrenar y otra para test en algunos casos, queden bien distribuidas las muestras de contacto y no contacto, en cada fase. De este modo, se asegura que no se utilicen una mayoría de muestras de un tipo frente a otro, por ejemplo de contacto frente a no contacto en la fase de entrenamiento. En este paso se debe tener cuidado con lo que se quiere barajar. En este caso, cada imagen individual es una fila de la matriz como se indicó anteriormente, por lo que solo se deben barajar las filas y no las columnas. A su vez, se deben barajar de la misma forma las muestras y sus etiquetas para que después no se entremezclen y se asigne un 0 a una imagen de contacto, por ejemplo. Esto se puede lograr de manera relativamente sencilla ya que la librería permite incluir diferentes variables en la misma función y aplicarles el mismo barajado.

Dependiendo del experimento llevado a cabo, en algunos casos se han utilizado técnicas de validación cruzada (cross-validation) haciendo uso de funciones de la propia librería, y en otros casos se han implementado manualmente estas técnicas, generando código para llevarlas

a cabo.

Cross-validation consiste en una técnica que se emplea para evaluar diferentes muestras sin que la parte del entrenamiento y la de test compartan datos. De esta manera se dividen las muestras existentes en un número elegido de grupos. En los experimentos en los que se utilizó esta técnica, se ha optado por dividir el conjunto de datos en 5 grupos. De esta manera, se entrena el algoritmo con todas las muestras de 4 grupos y se reserva uno de los grupos para test. La parte de test va rotando hasta que se realizan todas las combinaciones posibles. Se puede ver un ejemplo esquemático de este proceso en la Figura 4.3.

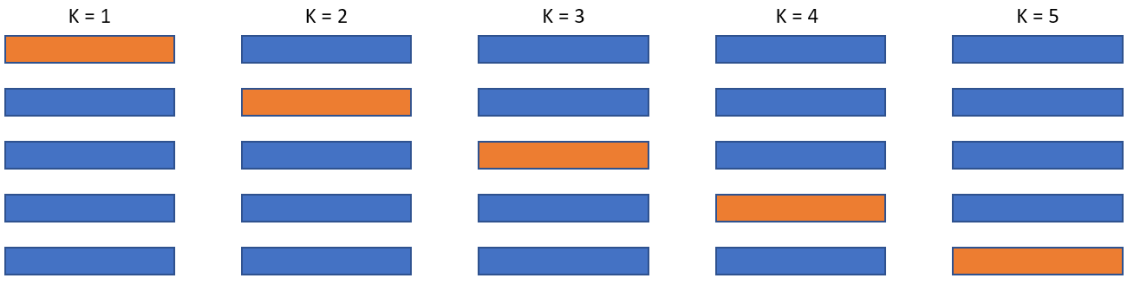


Figura 4.3: Representación simplificada de la división en la validación cruzada. Los rectángulos azules representan los grupos utilizados para el entrenamiento mientras que los naranjas representan las muestras usadas para test.

De este modo, se asegura que las muestras de test y de entrenamiento pertenecen a conjuntos disjuntos, y por lo tanto no se entrena y testea con muestras repetidas en ningún caso. Además, esta técnica permite asegurar que la elección del conjunto de test se realiza de manera arbitraria y no escogiéndola de modo que favorezca obtener resultados de detección altos.

Al usar las funciones de cross-validation de la librería, se puede escoger el número de divisiones y ciclos de las muestras y la salida que se quiere obtener con ellas. En este caso se especificaron las salidas con las métricas de evaluación anteriormente vistas, exactitud (accuracy), exhaustividad (recall) y precisión (precision). La librería permite devolver los valores calculados para cada una de las cinco iteraciones, de acuerdo con la división del conjunto de datos usado con la validación cruzada, por lo que se debe calcular el promedio de las cinco iteraciones, por lo general, manualmente.

Scikit es una librería muy útil y cómoda, sin embargo, en ocasiones puede resultar un tanto

opaca ya que es difícil observar los cálculos que realizan las funciones internamente. A su vez, al ser funciones fijas, no permite introducir parámetros ni cambios durante el proceso. Debido a esto, en otros experimentos se utilizó un código propio de desarrollo más manual en vez de rutinas y funciones ya implementadas en la librería, para poder entender, comprobar y controlar los cálculos y resultados obtenidos.

En estos experimentos, se realizaron los pasos previos igual que anteriormente, tanto la lectura de los datasets como el barajar las filas de la matriz de muestras. A continuación, se entrenó el algoritmo con las muestras de entrenamiento y se predijo con la función de test, sin utilizar la función de validación cruzada. El test se realizó a partir de las muestras reservadas para él, recién barajadas y esto se realizó durante cinco repeticiones, de manera que se obtenían valores diferentes de test con los que obtener un promedio.

Una vez realizada la prueba, se contrastaron las muestras predichas con las etiquetas reales de ellas, utilizando una matriz de confusión con la que poder ver el número de muestras correctas e incorrectas. A partir de la predicción y las etiquetas también se calcularon las métricas de evaluación enumeradas anteriormente.

Como se puede observar, Scikit Learn es una librería muy útil dentro de este tipo de aplicaciones resultando una muy buena herramienta que permite realizar y comprobar diferentes cálculos. Sin embargo, es una herramienta demasiado compacta, por lo que en ocasiones es difícil controlar el proceso de manera manual o dar flexibilidad de implementación al usuario y/o programador. No obstante, se puede hacer notar que igualmente también provee las funciones y rutinas necesarias para hacer de la implementación un proceso más manual, es decir más dirigido por el programador, dotándole de mayor flexibilidad para manejar parámetros, por lo que en ocasiones puede ser de interés, escoger adecuadamente las funciones y/o rutinas que presenta esta librería para implementar métodos basados en estos algoritmos de aprendizaje de un modo más compacto o proporcionando más posibilidades de configuración al programador.

4.2.1 Diagramas de flujo de Scikit Learn

A continuación se muestran dos diagramas de flujo donde se resume por pasos los códigos del primer y segundo experimento (Figura 4.4) y los del tercer y cuarto experimento

(Figura 4.5). Los códigos son muy similares pero existen algunas diferencias a la hora de implementarlos como se ha comentado en el apartado anterior. Estos experimentos se verán más profundamente en la Sección 5.2.2.

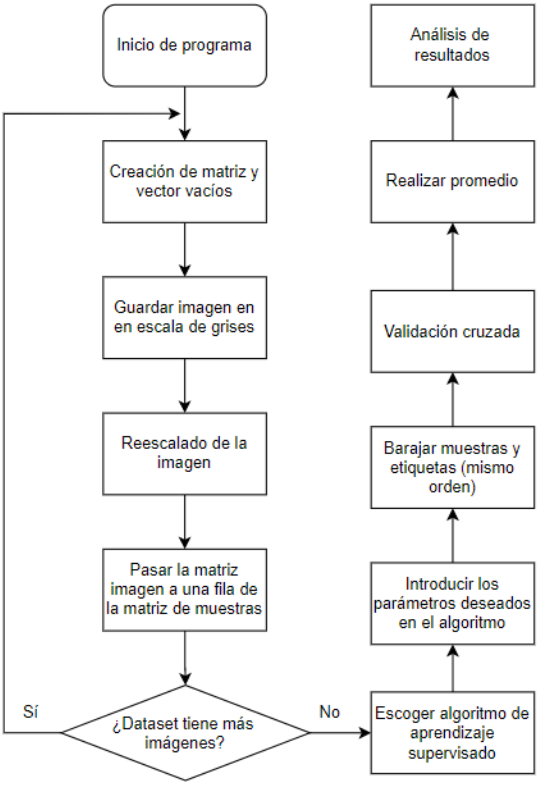


Figura 4.4: Diagrama de flujo para código de los experimentos uno y dos realizado con Scikit Learn.

4.3 Método propuesto en OpenCV

En los experimentos realizados con Scikit Learn, OpenCV se utiliza únicamente como herramienta de procesado de imagen. Sin embargo, aunque no es su principal cometido, también puede utilizarse para implementar algoritmos de aprendizaje supervisado.

En un primer momento de contacto con este tipo de algoritmos se utilizó OpenCV para procesar los datasets mediante las técnicas de aprendizaje propuestas en este trabajo, sin embargo, el costo de programar, en términos de esfuerzo, este tipo de técnicas es mucho mayor en comparación con Scikit Learn, por lo que se terminó por utilizar esta opción únicamente

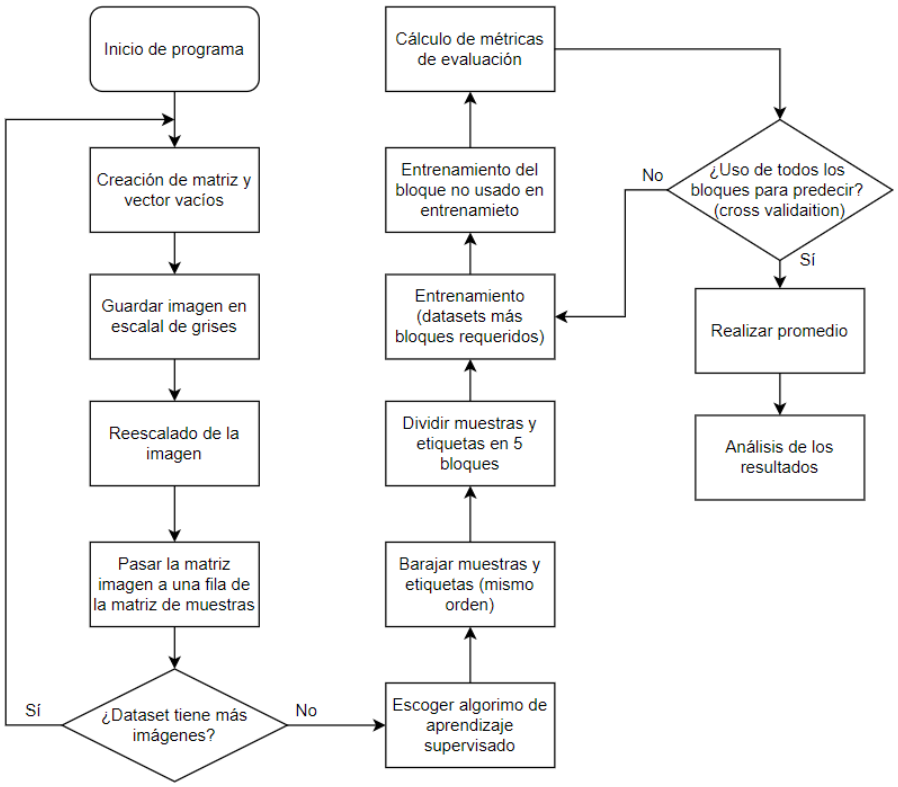


Figura 4.5: Diagrama de flujo para código de los experimentos tres y cuatro realizado con Scikit Learn.

para obtener algunos de los resultados.

Aun siendo de esta manera, debido a estas pruebas iniciales y que, realmente es una herramienta que estaba disponible para implementar técnicas básicas de aprendizaje, se realizó el código del primer experimento también utilizando OpenCV para comprobar que los resultados eran similares a los obtenidos con Scikit Learn y, por lo tanto, no eran muy dependientes de la librería software empleada.

La toma de imágenes de los datasets utilizados se realiza de la misma manera que con el método anterior, teniendo en cuenta que además ambos utilizan OpenCV.

Una vez obtenida la matriz de muestras y el vector de etiquetas, se realiza un barajado para entremezclar las muestras. En OpenCV, es necesario guardar el estado que se va a utilizar de aleatoriedad para poder aplicarlo tanto a las muestras como a las etiquetas debido a que solo se puede barajar una variable a la vez.

Respecto a la técnica de validación cruzada, OpenCV no tiene implementado este sistema

en su librería, por lo que se ha tratado de simular el proceso de manera manual. El número de muestras totales y las etiquetas se dividen en cinco variables distintas de muestras y otras cinco para las etiquetas. Una vez hecha la división, se repite el mismo esquema cinco veces, se concatenan cuatro variables que se utilizan para el entrenamiento y la quinta sin usar se utiliza para la validación del sistema.

Por cada iteración realizada, se calculan manualmente los datos de la matriz de confusión, debido a que OpenCV tampoco incorpora este tipo de cálculo. Para esto se comparan los resultados del aprendizaje automático con las etiquetas reales y se guardan las etiquetas correspondientes en caso de que sean positivos o negativos, falsos o verdaderos.

Para finalizar, se calculan las métricas utilizadas anteriormente en el método de Scikit Learn. Igual que sucedía con la matriz de confusión, el cálculo de las métricas no viene incluido en la librería, teniendo que implementarse las rutinas necesarias para poder calcularse. Por lo tanto, la implementación de las métricas corre a cargo del programador, es decir del proyectante en este caso.

Por último, se hace un promedio de las cinco iteraciones para observar los resultados globales.

4.3.1 Diagramas de flujo de OpenCV

De igual forma que con Scikit Learn, se adjunta a continuación un diagrama de flujo (Figura 4.6) donde se puede ver de forma general los pasos seguidos en el código. Se verá más sobre este uso en la Sección 5.2.2.

4.4 Mecanismos de evaluación

La obtención del modelo a partir del entrenamiento es la base para que funcione, pero un paso muy importante es la correcta comprobación de los resultados obtenidos. La utilización de diferentes métricas para evaluar el sistema es interesante para evitar que, resultados que a priori parecen buenos, realmente den problemas al comprobarlos mediante comparación con otras métricas. La idea es que el resultado no sea dependiente de la métrica usada.

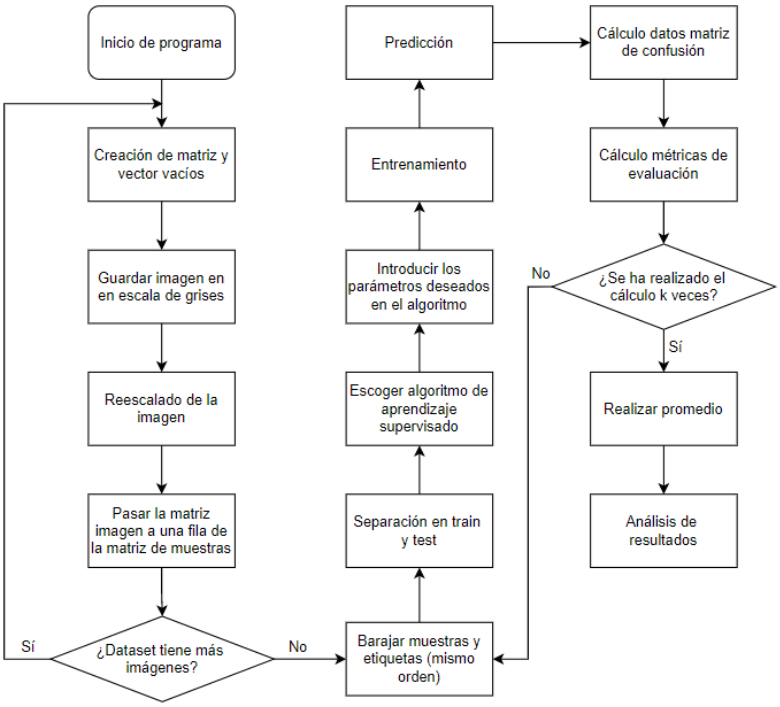


Figura 4.6: Diagrama de flujo para código realizado con OpenCV.

4.4.1 Matriz de confusión

La matriz de confusión es un método con el que se pueden visualizar de manera sencilla los aciertos y errores que puede haber al comparar las respuestas obtenidas con las que realmente deberían ser.

Como su propio nombre indica, los resultados se representan en formato de matriz, con un tamaño de dos filas y dos columnas. En cada celda se coloca un valor en concreto, según sea true positive (TP), true negative (TN), false positive (FP) o false negative (FN):

- True positive, TP: consiste en la obtención de un valor positivo cuando el resultado real es positivo.
- True negative, TN: el valor resultado es negativo al igual que el valor real.
- False positive, FP: el valor obtenido es positivo, mientras que realmente debería ser negativo.
- False negative, FN: se obtiene un resultado negativo pero el valor real es positivo.

En la Figura 4.7 se puede observar cómo se muestra la salida de la matriz y cómo sería en el caso del proyecto utilizando Scikit Learn, siendo ”O” el valor obtenido y ”R” el real.

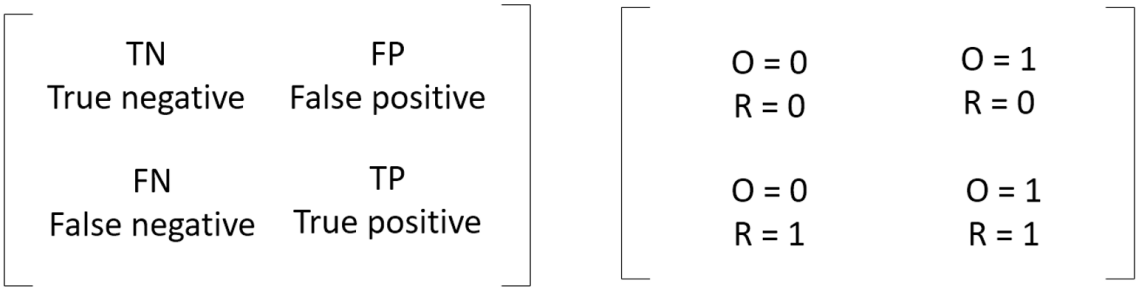


Figura 4.7: Esquema de salida de la matriz de confusión implementada con Scikit Learn.

Observando la forma de representación, se puede comprender que los valores que se quieren obtener son la mayor cantidad de valores verdaderos (TN y TP), minimizando los valores falsos (FN y FP), que son las imágenes que clasifica erróneamente el modelo.

4.4.2 Métricas de evaluación

Existen diferentes métricas de evaluación para determinar la eficacia de los algoritmos empleadas. En este proyecto se han escogido varias ya que, si solo se utilizara una, no se podrían analizar varios factores ya que cada métrica es diferente y al utilizar distintas se complementan haciendo la evaluación más eficaz. En este caso se han escogido tres de las métricas más utilizadas: exactitud (accuracy), exhaustividad (recall) y precisión (precision).

La medida de “accuracy” corresponde al porcentaje de elementos que han sido clasificados correctamente.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Esta medida puede dar problemas en el caso en que el dataset no sea equilibrado, es decir, que los conjuntos de categorías empleados no estén balanceados, mostrando un número distinto de muestras. En caso de que el modelo clasifique todas las muestras con la misma etiqueta, si el dataset tiene, por ejemplo, un 75% de muestras con esa etiqueta, se obtendrá un resultado de acierto del 0.75, cuando en realidad el modelo trabaja de forma errónea.

“Recall” da el porcentaje de aciertos positivos respecto del total de los positivos reales que

debería haber.

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

"Precision" es el porcentaje de aciertos positivos respecto de todas las clasificaciones positivas del modelo.

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Estas tres métricas pueden implementarse utilizando funciones propias de la librería de Scikit Learn. En el caso de OpenCV es necesario programar las rutinas para realizar los cálculos debido a que la librería no contempla la comprobación mediante estas métricas.

5 Experimentación y análisis de los resultados

En esta sección se describirá el procedimiento implementado para llevar a cabo agarres con la pinza Robotiq en la que se han montado los sensores DIGIT. Además, se detallarán los pasos llevados a cabo para adquirir imágenes táctiles, así como los métodos para determinar información de contacto entre objeto y pinza. Finalmente, se expondrán los resultados obtenidos llevados a cabo en diversos experimentos. Todos estos ensayos tienen como objetivo determinar si el uso de DIGIT es adecuado para la tarea de detección de contacto en tareas de agarre, y si proporcionan la fiabilidad necesaria.

5.1 Implementación del sistema de percepción táctil

5.1.1 Pinza

El modelo de la pinza 2f_140 puede descargarse a partir del repositorio oficial, en GitHub, de la marca Robotiq. Con estos archivos se dispone de las rutinas necesarias que permiten llevar a cabo ordenes que faciliten realizar el movimiento de la pinza tanto en el entorno simulado como en la pinza física.

Para realizar el movimiento es necesario descargar los archivos en un "workspace" de ROS. Para ello se utiliza la herramienta "catkin_make" y la carpeta descargada será un paquete de ROS listo para funcionar.

- Entorno simulado

Una vez descargado el repositorio mencionado anteriormente, se pueden utilizar una serie de comandos para visualizar la pinza.

Se puede cargar la visualización dentro del paquete de Robotiq mediante la función *roslaunch robotiq_2f_140_gripper_visualization test_2f_140_model.launch*. Se utiliza el en-

torno de Rviz para cargar la salida en la que se puede ver el modelo de la pinza o las colisiones de esta como en la Figura 5.1.

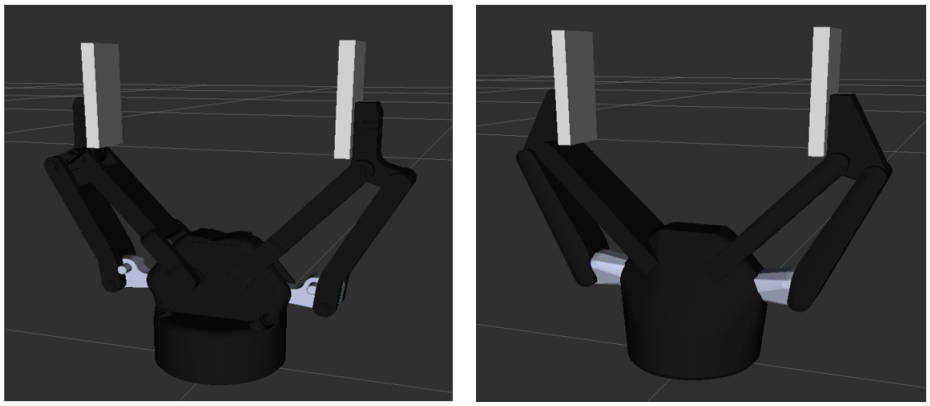


Figura 5.1: Visualización de la pinza según su modelo visual mecánico y el modelo de colisiones.

Además, mediante las carpetas proporcionadas por la empresa Robotiq se pueden recrear trayectorias simuladas mediante la herramientas RViz.

Para ello es necesario un archivo de tipo urdf que pueda organizar los diferentes archivos propios de la pinza. En este caso, este archivo de configuración fue proporcionado por los componentes del grupo de investigación AUROVA ya que ya lo tenían en parte preparado para utilizar en proyectos de investigación como COMMANDIA. No obstante, el fichero proporcionado, venía no solo con el diseño CAD de la pinza sino también con el diseño de las partes de otros robots y objetos de un complejo entorno industrial. Así, que se analizó las partes para eliminar todas aquellas innecesarias del archivo, que no correspondían a la pinza, y únicamente visualizar esta y permitir el correcto funcionamiento de la aplicación. Además, se utilizó este fichero ya limpio, para crear un archivo MoveIt! (Coleman y cols., 2014) que permitiera la planificación y control de los movimientos de la pinza.

MoveIt! consiste en una herramienta de software que se puede utilizar dentro del entorno de ROS para realizar simulaciones de brazos y otros componentes robóticos.

Para crear el paquete de MoveIt! se debe acceder al workspace creado donde se encuentran los archivos de la pinza y el archivo urdf. Es importante recordar ejecutar el comando ”catkin_make” seguido de *source devel/setup.bash* para actualizar el workspace. Una vez realizado este paso previo, se debe abrir la aplicación, lo más cómodo es mediante el terminal,

y se escoge crear un nuevo paquete a partir del archivo "urdf" mencionado.

Tras abrir la herramienta se pueden comenzar a definir las características de la pinza. Para esto deben irse seleccionando las diferentes pestañas que aparecen en la interfaz para completar el modelado de los puntos de interés de la pinza.

Debe tenerse en cuenta que, con esta aplicación, se suelen definir brazos robóticos, desde un origen a un punto final de estos. En el caso de la pinza es similar, se debe colocar el punto inicial de la cadena en uno de los dedos y trazar la cadena hasta el punto contrario del otro dedo. Esto se comprobó tras tratar de entender la pinza como una cadena que se originaba desde la base y se separaba hacia los dos dedos. Esta configuración daba problemas ya que el programa no reconocía la pinza correctamente y se simulaba el movimiento de la pinza de manera que solo se cerraba uno de los dedos o se soltaban partes de esta. Así, con estos cambios implementados, cada uno de los dedos se unen en un punto como una cadena cinemática única, de modo que el movimiento entre los dos es acoplado. Esto hay que hacerlo porque los motores de la pinza están sobreactuados, es decir hay más articulaciones que motores para mover los dedos.

Tras definir los diferentes aspectos de la pinza como las articulaciones activas o los grupos de movimiento se pueden crear posiciones para la pinza que se utilizarán posteriormente en la simulación como puntos a alcanzar.

Una vez creado el paquete MoveIt!, se puede utilizar el archivo creado en él, *demo*.launch. Con este archivo se puede abrir la simulación de la pinza en RViz y controlar los movimientos. Para lanzar el archivo simplemente se debe hacer ejecutar el comando del archivo con *roslaunch <nombre de la carpeta de MoveIt!> demo.launch*.

Una vez abierto el programa se puede acceder a la pestaña "planning" y en sus apartados escoger la posición en la que se quiere comenzar y/o terminar el movimiento. Tras escoger esto, se puede escoger la opción "plan", donde se puede ver qué movimiento haría mediante una sombra o "plan and execute", con el que se puede crear este movimiento previo y posteriormente mover la pinza a esa posición elegida. En este caso, las posiciones creadas anteriormente fueron la pinza en posición abierta y en posición cerrada, por lo que se probó a alternar entre estas opciones. Se pueden ver los diferentes estados de la planificación en la Figura 5.2.

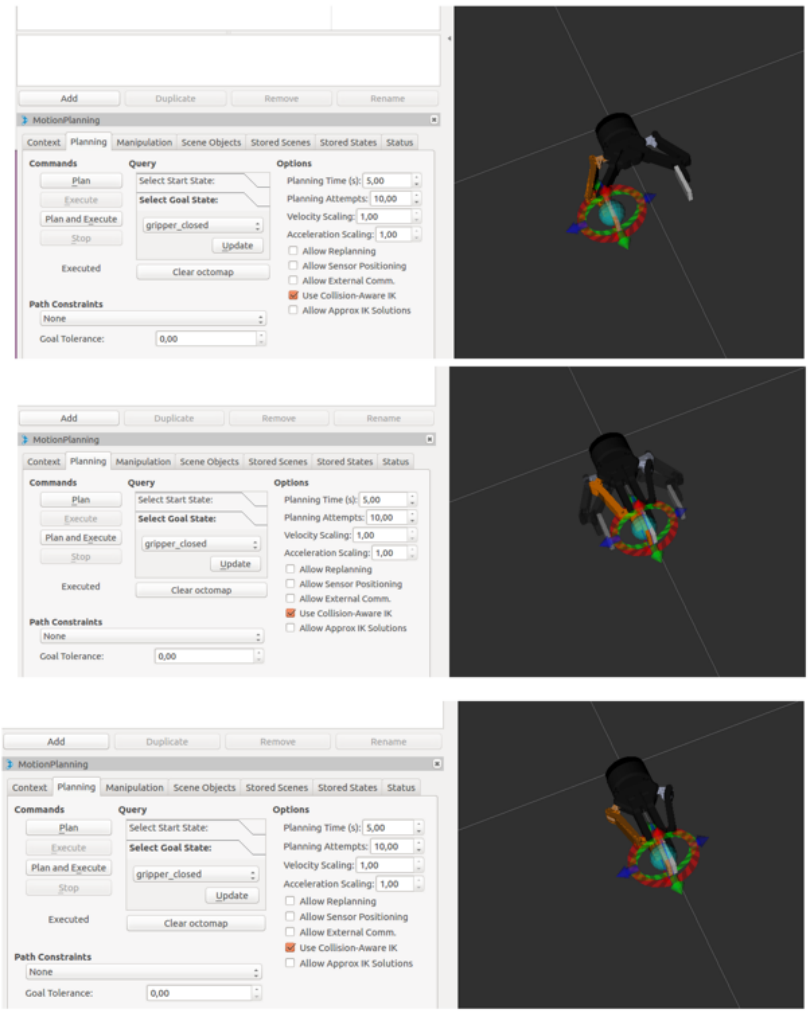


Figura 5.2: Movimiento de cerrar la pinza en simulación.

- Modelo físico

Para mover el modelo físico se utilizaron las instalaciones del laboratorio donde estaba el ordenador conectado mediante comunicación serie a la pinza física.

Para poder moverla, igual que en la simulación, es necesario descargarse los ficheros ofrecidos por Robotiq en el workspace que se quiere utilizar.

Para realizar el control de la pinza mediante ROS existen tutoriales en la página oficial para mover diferentes modelos. En este caso se utilizó un protocolo Modbus RTU para mover el modelo del laboratorio. El protocolo Modbus es una arquitectura que se basa en el paso de información maestro/esclavo, siendo en este caso el ordenador el maestro y la pinza el

esclavo. La parte RTU hace referencia a una Unidad de Transmisión Remota. Este sistema realiza envíos de paquetes de información para transferir mensajes de manera estandarizada según el protocolo.

Para enviar el driver encargando de controlar el movimiento de la pinza se utiliza el comando *rosrun robotiq_2f_gripper_control Robotiq2FGripperRtuNode.py /dev/ttyUSB0*. En otro terminal se lanza el nodo de control que envía mensajes al nodo de la pinza, que espera la llegada de estos. Para ello se utiliza el comando *rosrun robotiq_2f_gripper_control Robotiq2FGripperSimpleController.py*. Con este comando aparecerán una serie de opciones para utilizar para poder mover la pinza desde el ordenador, se pueden ver en la Figura 5.3.

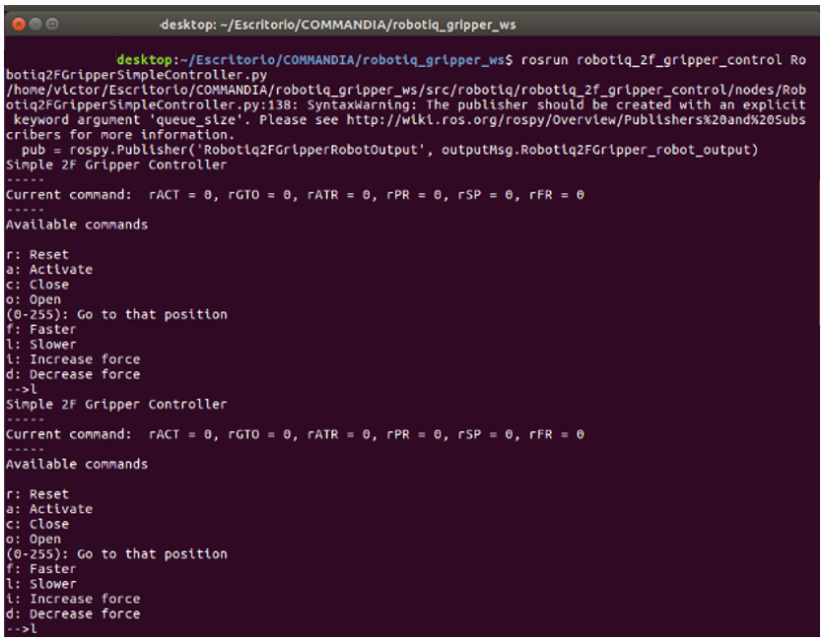


Figura 5.3: Salida por terminal para ejecutar el movimiento de la pinza.

Una vez que se tiene la lista de comandos lo primero que se debe hacer es pulsar ”r”, para hacer Reset. Este comando pone las variables a 0 inicialmente para poder iniciar el proceso. A continuación, debe pulsarse ”a”, Activate, de manera que se calibra la pinza, primero abriéndola del todo para después cerrar y volver a abrirse. Debido a los sensores sobresalen un poco por fuera de los dedos de la pinza, en este momento se colocaba una esponja, de manera que al cerrarse la pinza para calibrar, no se golpeasen dañándose los geles. Esto puede verse en la Figura 5.4.

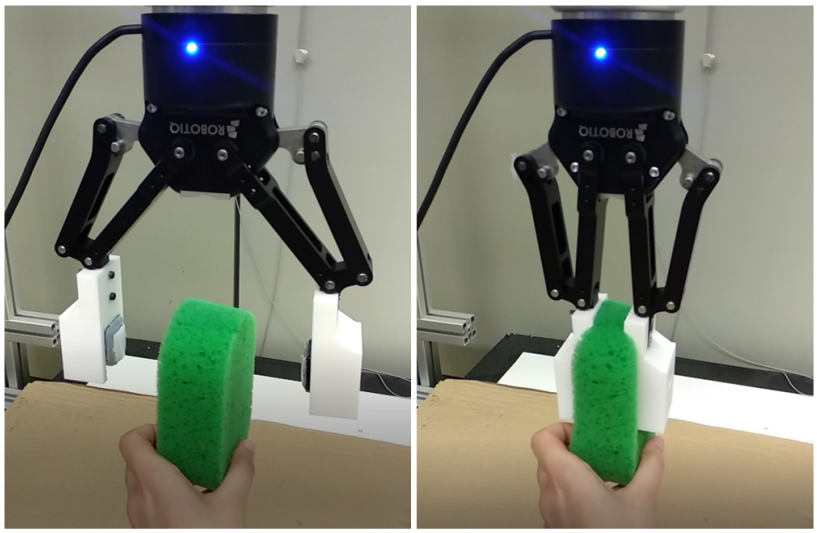


Figura 5.4: Calibración de la pinza.

Una vez realizado este proceso ya puede usarse la pinza con el cierre que se quiera, colocando ”o” (open) para abrir completamente la pinza o ”c” (close) para cerrarla. Si se quiere abrir o cerrar mediante puntos intermedios se puede introducir un número de 0 a 255 indicando los grados de cierre o apertura que se quieran.

Además, existen otros valores que se pueden introducir como ”f” para que vaya más rápido, ”l” para reducir la velocidad de la pinza o ”i” y ”d” para aumentar o disminuir la fuerza que ejerce respectivamente. Se recomienda disminuir la velocidad en un primer momento ya que para realizar las pruebas y colocar los objetos es preferible que la pinza no se mueva demasiado rápido para poder evitar posibles accidentes al colocar los objetos de los ensayos.

5.1.2 Sensor

Para la utilización de los sensores DIGIT se necesitan una serie de paquetes descargables desde el repositorio oficial del sensor, cuyo diseño pertenece a Facebook. Una vez que se tienen disponibles se puede comenzar a utilizarlos usando las funciones ya implementadas en la librería.

Existen ciertas funciones relacionadas con la toma de imágenes de manera indirecta como puede ser el cambio de intensidad de los LEDs, la resolución o la velocidad de imágenes por segundo que se pueden tomar. No es necesario utilizarlas inicialmente debido a que el sensor

viene con estas características establecidas de manera estándar, y no parece que requieran un ajuste inicial, por lo que se ha optado por emplear la configuración por defecto que es la recomendada.

A mayores existen otras funciones incluidas en la librería que se pueden utilizar para la toma de imágenes para el dataset, como puede tomar el frame mostrado y/o guardarlo en la ubicación que se desee.

Para probar estas funciones en un primer contacto se pueden descargar algunos ejemplos del repositorio oficial de Facebook.

Al probar uno de estos códigos por primera vez para observar la salida se apreció que la imagen salía cortada o desplazada, sobresaliendo alguna de las partes por el lado contrario. Esto puede ocurrir al utilizar una máquina virtual, la cual puede dar problemas para visualizar las imágenes. En este proyecto, los datasets se tomaron en el laboratorio sin utilizar ninguna máquina virtual por lo que todas las imágenes tienen la calidad esperada sin problemas a la hora de guardarlas.

5.2 Sistema de reconocimiento táctil

5.2.1 Datasets utilizados

Para comprobar las capacidades del sensor se realizaron diferentes experimentos. Con ellos se pretende determinar la precisión dentro del sensor para algunas tareas sencillas de reconocimiento táctil. Para hacer el estudio de precisión, se han elaborado varios "dataset" (bases de datos de muestras) compuestos por imágenes táctiles de distintas unidades de sensor táctil, en las que se ha registrado distintos contactos de operaciones de agarre y contacto con distintas superficies de objetos. El objetivo es estudiar el rendimiento del sensor para ser empleado en reconocimiento táctil en tareas de agarre robótico. Para llevar a cabo el estudio, se ha desarrollado un método de reconocimiento en el que se implementan distintos algoritmos que hacen uso de técnicas de aprendizaje automático.

A continuación, se presenta una breve descripción de los objetos empleados para generar los contactos en tareas de agarre, que compondrán los dataset. Los datasets se han desarrollado en colaboración con miembros del grupo AUROVA, como parte de este TFG y enmarcados

dentro de algunos desarrollos parciales del proyecto europeo COMMANDIA que dirige mi tutor de TFG, proyecto que está enfocado a manipulación de objetos deformables y donde se emplea percepción táctil.

- Objetos utilizados para generar las imágenes táctiles

Para cada experimento se ha elaborado un dataset específico. En cada uno, se ha utilizado un conjunto concreto de diferentes objetos. Con ellos se pretende probar la capacidad de detección de contacto en los agarres que pueden ofrecer los sensores tacto-visuales construidos.

Así, en los dos primeros experimentos, estando los dos relacionados, se utilizaron los mismos objetos para crear dos datasets de imágenes táctiles distintos. Éstos se comentarán a continuación.

En la Figura 5.5, tomada en laboratorio de AUROVA, se pueden observar cinco objetos que pertenecen a una base de datos más amplia creada por la Universidad de Yale (Calli y cols., 2017), y cuyos objetos fueron facilitados por esta universidad americana al grupo AUROVA para que sirvieran como "benchmarking" en la comparativa de algoritmos y resultados. Por lo tanto, este conjunto de objetos fue diseñado por Yale, para poder facilitar una base común de pruebas en proyectos e investigaciones en el área de la manipulación robótica. De este modo, se asegura una manipulación en condiciones similares, lo que favorece una comparativa de resultados justa.

Los objetos escogidos en el primer grupo, de entre todos los de la base de datos de Yale, han sido una pelota de tenis y otra de béisbol, un bote de mostaza, una taza y un recipiente de producto para limpiar ventanas.

El segundo grupo de objetos del que se compone el dataset utilizado en los experimentos, está formado por otros cinco objetos que se puede ver en la Figura 5.6. Estos objetos pertenecen a una base de datos de objetos propios del grupo AUROVA de la Universidad de Alicante. Entre estos objetos se encuentran un recipiente de jabón, un ambientador, una lata de Aquarius, una caja de madera y una caja de Aquoral Forte.

Estos diez objetos, por lo tanto, se utilizaron en los datasets de los primeros dos experimentos, tomando muestras de ellos tanto en casos de contacto como de no contacto. Cada una de las muestras consiste en una imagen táctil adquirida a partir del sensor C, cuya carcasa

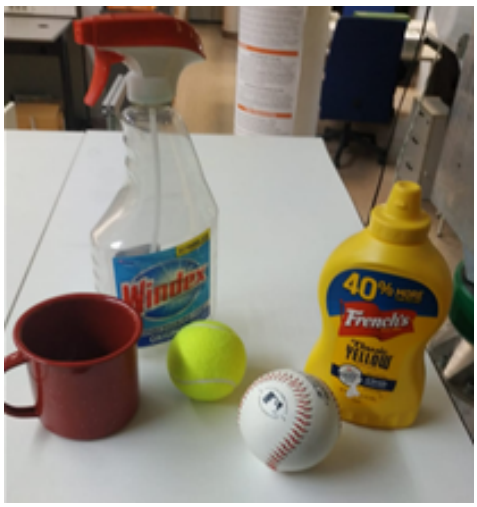


Figura 5.5: Imagen de los objetos del dataset de Yale utilizados.

Figura 5.6: Imagen de los objetos del dataset de AUROVA utilizados.

está fabricada con PLA.

Para el tercer y cuarto experimento se utilizaron imágenes táctiles registradas con tres unidades distintas de sensores DIGIT. Una de las unidades de DIGIT ya fue utilizada con el dataset anterior (experimentos uno y dos). Los otros dos son unidades distintas. Hay que recordar que la construcción manual de cada unidad de sensor DIGIT hace que estos sean únicos y no se puede encontrar dos unidades exactamente iguales. Por lo que se dificulta el objetivo de desarrollar un método que funcione correctamente en tareas de detección y reconocimiento con unas u otras unidades indistintamente.

Así, conviene hacer notar que uno de estas dos unidades, fue fabricada con carcasa enteramente ABS, y el otro sensor con partes de la carcasa ABS y otras partes como las piezas

frontales fabricadas con resina.

El dataset para el tercer y cuarto experimento está compuesto por imágenes táctiles de objetos de los datasets anteriores y de objetos nuevos. En concreto, de Yale se ha usado la pelota de tenis, de la Universidad de Alicante el recipiente de jabón, y además se han incorporado otros objetos nuevos algunos de Yale (bote de pringles, manzana) y otros de la Universidad de Alicante (spray ambientador y cuatro suelas de distintos materiales y formas). Estos objetos pueden verse en la Figura 5.7.

Figura 5.7: Imagen de los objetos del dataset usado en el tercer y cuarto experimento.

- Imágenes táctiles de los datasets

El primer experimento utiliza un dataset con imágenes obtenidas de una unidad de un sensor DIGIT cuya carcasa se ha construido a partir de material de impresión PLA amarillo. Para identificar correctamente este sensor tanto en la memoria como en la experimentación, a partir de ahora se denotará como unidad de sensor C. Este dataset contiene un total de 2960 imágenes táctiles.

En el dataset empleado para el primer experimento, se registraron imágenes táctiles obtenidas por los contactos y aproximaciones entre sensor y objeto, a partir de zonas de la superficie de los objetos en las que se aseguraba que no se generaba ningún tipo de textura. Conviene hacer notar que, al ser el fundamento de construcción de la unidad del sensor, la tecnología óptica no produce el mismo tipo de imagen táctil si el sensor entra en contacto con el objeto en una región de color homogéneo de la superficie de éste (no habrá casi textura) o si lo hace a partir de una región donde existe una serigrafía o un rótulo (habrá textura), por poner un ejemplo. Por lo tanto, en algunos objetos la muestra de contacto que se registra es

distinta si se varía la pose del objeto o la zona de agarre.

En este dataset, se encuentran imágenes sin ningún tipo de textura concreta a primera vista o al menos no muy pronunciada. Se pueden ver un par de muestras del dataset en la Figura 5.8, un frame corresponde al sensor realizando un agarre y el otro al sensor en un estado de no contacto.

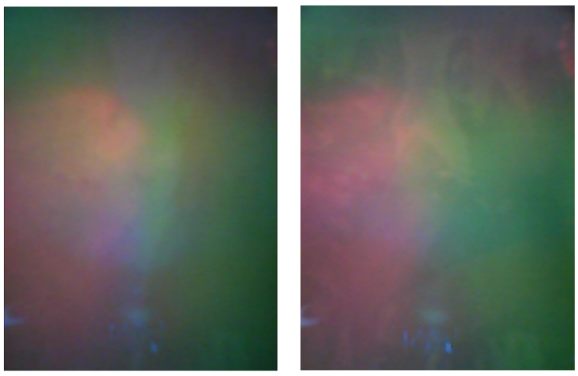


Figura 5.8: Ejemplos del dataset de imágenes sin textura en estado de agarre (izquierda) y sin contacto (derecha).

El segundo experimento utiliza imágenes táctiles de los mismos objetos, pero en esta ocasión los agarres, pueden generar texturas como se aprecia en el letras de las etiquetas de los objetos utilizados (Figura 5.9). Estas imágenes también se obtuvieron utilizando el sensor C. El total de imágenes de este conjunto es de 1990 imágenes.

Figura 5.9: Ejemplos del dataset de imágenes con textura en estado de contacto (izquierda) y de no contacto (derecha).

Los últimos dos experimentos (experimento tres y cuatro) utilizan un dataset que se divide

en tres conjuntos más pequeños. Cada conjunto está formado por imágenes táctiles adquiridas a partir de cada uno de los distintas unidades de sensores táctiles, como se comentaba en el apartado dedicado a las imágenes. El sensor A es el sensor fabricado en carcasa ABS, el sensor B cuyas piezas de la parte posterior y medio son en ABS y su parte posterior en resina y el sensor C fabricado en PLA. El montaje de las piezas de B formadas por distintos materiales se debe simplemente a corrección en la operativa de ensamblaje, ya que de esta manera encajaban mejor unas sobre otras.

Cada uno de estos tres conjuntos individuales se toma con uno de los sensores, de manera que se tengan claramente identificadas las muestras de cada sensor y así después se puedan combinar para realizar diferentes pruebas, tal y como se verá en el capítulo de experimentación.

Este dataset consta de tres conjuntos, el correspondiente al sensor A consta de 6874 imágenes totales, el correspondiente al sensor B contiene 4493 imágenes y por último el del sensor C tiene un total de 4208 imágenes.

Algunas imágenes de los datasets del primer y segundo experimento pueden encontrarse en en el conjunto de muestras correspondientes a C que se han usado en este nuevo dataset, donde aparecen también de manera disjunta muestras de las otras unidades de sensor, A y B. En cualquier caso, conviene destacar, que en este dataset el conjunto de muestras de C se ha aumentado con nuevas imágenes hasta alcanzar las 4208 imágenes.

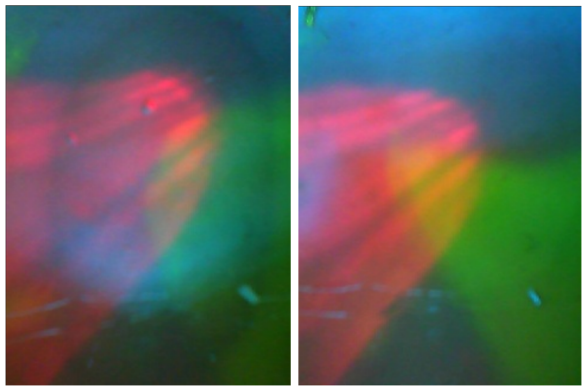
En la Figura 5.10 se pueden observar muestras de las imágenes del dataset pertenecientes a cada uno de los tres conjuntos, tanto de contacto como de no contacto.

5.2.2 Pruebas realizadas

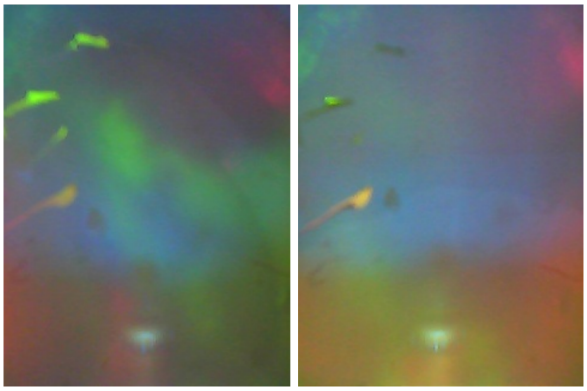
Para comprobar el alcance del sensor a la hora de clasificar el agarre en contacto y no contacto se han realizado cuatro experimentos distintos.

- Primer experimento

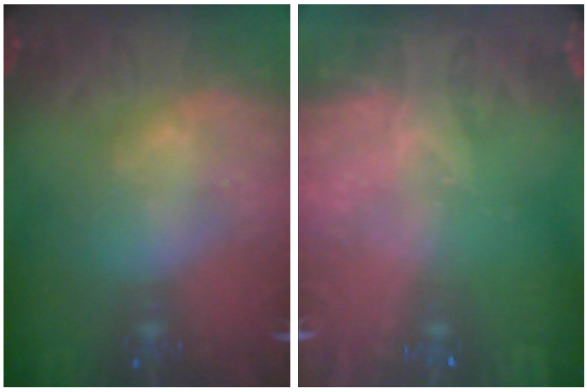
En el primer experimento se utilizó un dataset con imágenes sin texturas como se ha visto en la Sección 5.2.1. Para este banco de imágenes se propusieron tres algoritmos: KNN, RF y SVM. A su vez, dentro de las pruebas de cada algoritmo se probaron a variar algunos



(a)



(b)



(c)

Figura 5.10: Muestras de los datasets correspondientes a los sensores A (a), B (b) y C (c).

parámetros, como el número de árboles o el número de vecinos consultados, para ver cómo estos cambios afectaban al resultado final y encontrar la mejor combinación posible para el modelo de predicción.

Para los resultados de los experimentos se utilizaron las tres métricas vistas anteriormente:

exactitud (accuracy), exhaustividad (recall) y precisión (precision). En algunos casos y con algunas métricas, es el caso de precisión, se han obtenido resultados de 100%, es decir un promedio de 1.0 con desviación típica 0.0 en una escala de [0,1]. Esto se debe a que en las diferentes muestras no se obtenían valores de falsos positivos en la detección, y por tanto se alcanzaba la máxima precisión. Por esta razón, y en general, se ha decidido no incluir los valores de esta métrica en la medida de rendimiento de los métodos, puesto que no permitían discernir diferencias de predicción entre modelos. En caso de haber algún resultado diferente se especificará en el apartado correspondiente.

En la prueba con el algoritmo KNN se probó a variar el número de vecinos de las muestras entre valores de 1, 3, 5 y 7. Los mejores resultados es obtuvieron con las dos primeras combinaciones, disminuyendo los promedios de exactitud y exhaustividad en el resto de los casos.

En el ensayo con el algoritmo RF se comprobaron los resultados cambiando el número de árboles utilizados. Los mejores resultados se dieron con 50 y 100 árboles, obteniendo resultados muy próximos a 1.0, con 0.999662 en exactitud (accuracy) y 0.999324 en exhaustividad (recall). Si se compara con el rendimiento medido en KNN se puede ver que RF obtiene unos resultados superiores por centésimas, por lo que ambos son muy próximos.

En el caso de uso del algoritmo SVM, se realizaron dos combinaciones de parámetros. Por un lado, se ajustó el parámetro C en 1.0 y se varió el tipo de kernel entre cuatro tipos posibles: lineal (linear), poligonal (poly), radial (RBF) y sigmoide (sigmoid), proporcionados por la biblioteca de Scikit Learn.

El kernel es la función con la que se separarán las diferentes muestras, siendo la forma que tomará el hiperplano para la clasificación de estas. El parámetro C afecta a la distancia entre la muestra más cercana al hiperplano y este mismo. De esta forma, si el valor de C es mayor, se escogerá un margen menor, pero será más estricto a la hora de separar las muestras, mientras que, con valores menores, se buscará un margen lo mayor posible, aunque esto signifique sacrificar algunas muestras clasificándolas fuera de su clase.

Debido a que los mejores resultados se obtuvieron con el kernel lineal y poligonal se escogió el primero por tener una mayor sencillez computacional y se varió el valor del parámetro C. En este caso, se escoge finalmente entre el parámetro C con valor 1.0 y 10000.0, debido a que

ambos resultados son iguales. Por sencillez se determina, para el algoritmo final, un kernel lineal y un parámetro C igual a 1.0.

La comparación de la variación entre los diferentes algoritmos y las combinaciones de los parámetros se pueden ver en la Figura 5.11 .

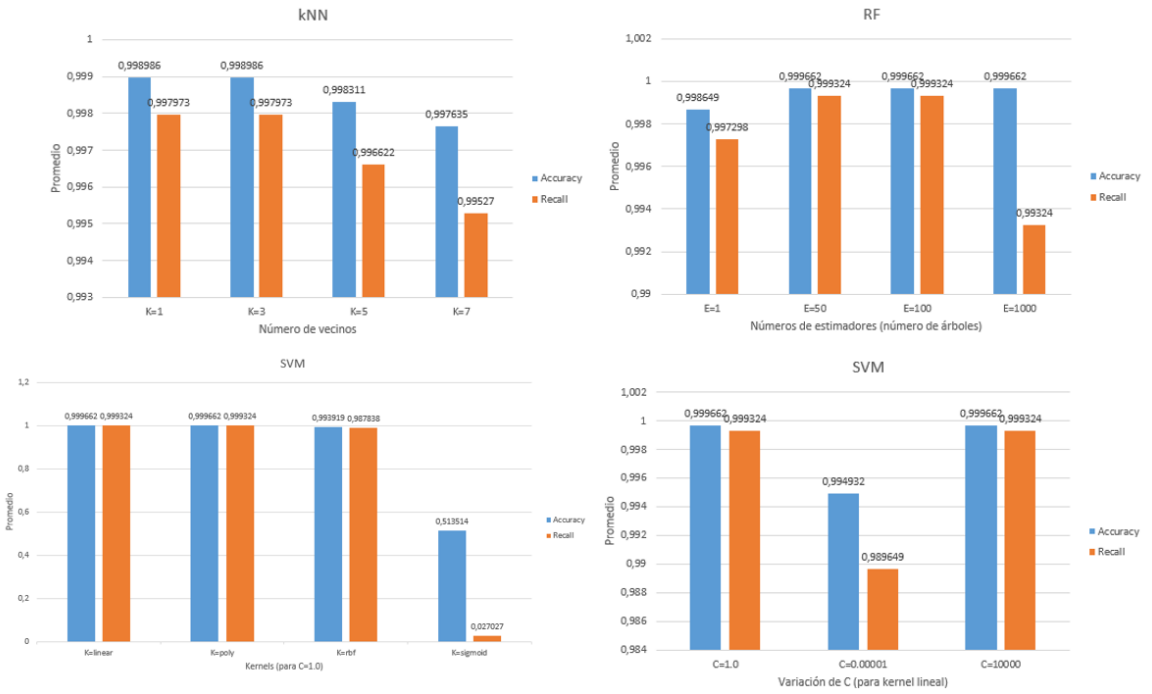


Figura 5.11: Resultados del primer experimento utilizando diferentes algoritmos.

Finalmente, comparando los tres algoritmos utilizados, se podría escoger como mejor método el que ha obtenido mejores resultados y que es más sencillo de ejecutar, siendo este el algoritmo RF con un número de 50 o 100 árboles.

En la toma de contacto con la programación de los algoritmos de aprendizaje supervisado se comenzaron a realizar los cálculo con OpenCV, debido a una mayor familiaridad previa con esta herramienta, adquirida durante la titulación. Sin embargo, como se comentó en la Sección 4.3, es una herramienta menos intuitiva y que implementa menos funciones para el aprendizaje supervisado, ya que está más enfocada a procesamiento de imagen y técnicas de visión por computador. Por estas razones, se han querido reflejar las diferencias y similitudes del uso de esta herramienta comparando los resultados obtenidos con Scikit Learn (versión 0.23.2) y los obtenidos con implementaciones que se han llevado a cabo en OpenCV (versión 4.5.1.48). Los

resultados de rendimiento alcanzados, empleando la misma técnica de predicción, es decir RF, con dos implementaciones que he desarrollado distintas para cada una de las dos librerías, se pueden observar en la Figura 5.12.

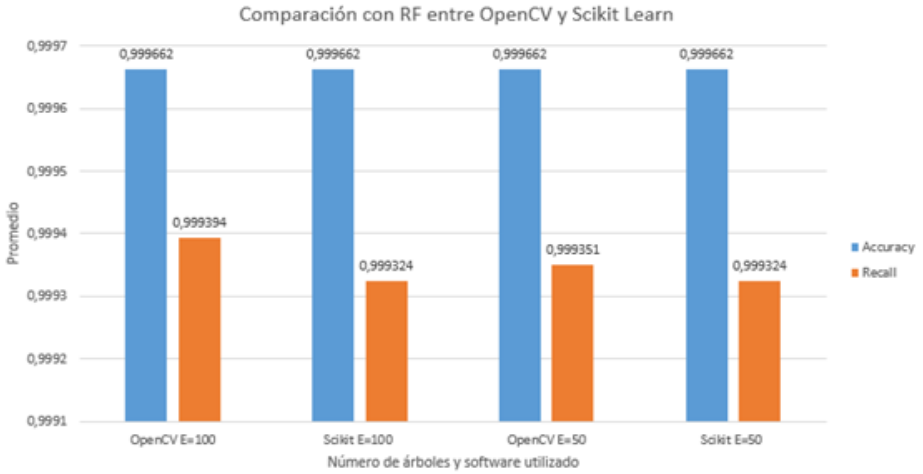


Figura 5.12: Comparación entre resultados obtenidos con Scikit Learn y con OpenCV.

Se pueden ver que los resultados son muy similares, existiendo una mínima variación en la evaluación de la exhaustividad (recall). De esta manera se pueden comprobar que los resultados son correctos y que se puede utilizar OpenCV para el aprendizaje supervisado si así se desea.

No obstante, se ha comprobado la comodidad del uso de Scikit Learn y poniendo en evidencia que OpenCV es una herramienta más incómoda en este tipo de algoritmos, teniendo que invertir un mayor tiempo en la programación de lo necesario en la mayor parte de los casos.

Como desventaja remarcar que, Scikit por contra es una herramienta muy cerrada, y en ocasiones no permite demasiada flexibilidad al programador. En estos casos es necesario ejecutar códigos más "manuales", es decir se requiere programar funcionalidades adicionales para tratar y organizar los datos, así como otras rutinas en el caso de que se desee realizar algún cálculo que no se contempla en las funciones predefinidas de esta herramienta, algo que se tuvo que hacer durante la realización de este proyecto.

- Segundo experimento

Para el segundo experimento se utilizó el dataset en el que las imágenes táctiles muestran texturas (ver Figura 5.9 de la Sección 5.2.1), en el que diversos objetos al entrar en contacto o por cercanía con el sensor, habían generado texturas debido a la heterogeneidad de la superficie del objeto (serigrafías de etiquetas, diversidad de etiquetado, etc.). En este caso se quiere observar, si los métodos implementados, son capaces de predecir correctamente el agarre en situaciones más complejas como ésta.

Para este experimento se utilizaron los mismos métodos de programación que en el ensayo anterior, para poder compararlos.

Al utilizar el algoritmo KNN se puede observar que el mejor resultado se encuentra al utilizar un solo vecino, decayendo el resto de los resultados. En el caso de utilizar tres vecinos, los resultados se igualan al anterior experimento.

En el caso de RF, se obtienen muy buenos resultados utilizando tanto 50, 100 o 1000 árboles, obteniendo resultados con un promedio de 0.999497 en exactitud (accuracy) y 0.998995 al observar la exhaustividad (recall).

Al utilizar SVM, igual que antes, se obtienen los mejores resultados al utilizar los kernels lineal y poligonal, obteniendo los mismos resultados, 0.999497 y 0.998995 en exactitud (accuracy) y exhaustividad (recall), respectivamente. Se puede observar en la Figura 5.13 que en el caso de la utilización de una función sigmoide como kernel, se obtienen resultados distintos, obteniendo valores bastante pobres en exactitud. En este caso, además, se obtuvo un promedio en la precisión de 0.5. Aunque el resultado de la exhaustividad parezca acertado, conociendo las otras dos métricas podemos descartar este kernel como adecuado para la clasificación.

De igual manera, se vuelve a comprobar que se obtienen mejores resultados en caso de tener un parámetro C mayor. Tanto C siendo 1.0 como 100000.0 se obtienen buenos resultados por lo que, de igual manera, se escogería finalmente el valor 1.0 por mayor sencillez.

La comparación entre los diferentes algoritmos en este experimento se puede ver mediante varias gráficas en la Figura 5.13.

Comparando ambos experimentos se puede observar que ambos obtienen resultados realmente altos, aproximándose de manera casi perfecta a 1.0. De cada mezcla realizada mediante

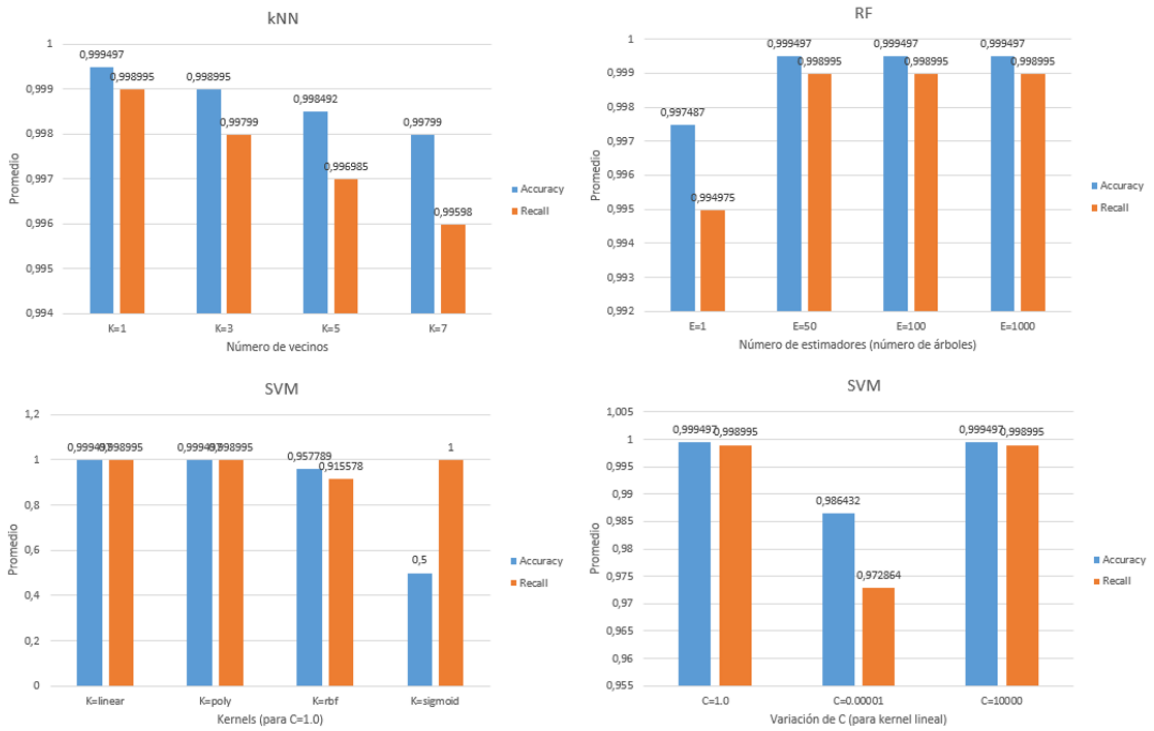


Figura 5.13: Resultados del segundo experimento utilizando diferentes algoritmos.

validación cruzada, una de cada cinco combinaciones obtenía un error, siendo el resto clasificaciones perfectas por lo que el promedio de todas ellas es muy alto, con una desviación típica muy próxima a cero en la mayor parte de los casos.

Sin embargo, aunque los mejores valores de cada algoritmo se aproximan a 1.0, el mejor resultado, teniendo en cuenta la mayor comodidad de programación del algoritmo en caso de empate, se obtiene con RF en el primer experimento, usando 50 o 100 árboles. Por tanto, se tendrá en cuenta RF con 100 árboles como modelo final y se utilizará en el resto de experimentos.

- Tercer experimento

El tercer experimento se subdivide en tres ensayos. En todos ellos se utilizan los datasets de las unidades de los sensores DIGIT, A y B (ver Sección 5.2.1), para entrenamiento, pero en cada uno de ellos varían la parte de test. El algoritmo de aprendizaje supervisado escogido es RF con 100 árboles debido a los buenos resultados obtenidos en los experimentos anteriores.

En el primer ensayo se entrena con todas las muestras del sensor B. Las muestras del sensor

A se dividen en 5 grupos. De manera que se puedan dividir en cinco grupos con el mismo número de imágenes por cada uno, algunos frames se descartan. A continuación, se realizan cinco combinaciones, de modo que la fase de entrenamiento se lleva a cabo con todas las muestras de B y con cuatro de los grupos de muestras en los que se ha dividido el conjunto de A. El 20% de muestras restantes de A se utilizan para el test. De este modo, lo que se consigue es entrenar con muestras más o menos balanceadas del sensor A y B. y testear la predicción con el sensor A. Esto, además, se repite en varias ocasiones, variando el 20% de las muestras de A usadas para test y siempre asegurando que el sistema nunca se entrena con muestras que se usarán en los tests. Esta técnica se conoce como validación cruzada y ha sido comentada en la Sección 4.2.

Se pueden observar que los resultados (Figura 5.14) son muy buenos con algún error entre ellos. Los valores de las métricas se encuentran por encima del 0.997 de promedio, observando las tres formas de evaluación. Se debe tener en cuenta que los resultados son variando, mediante validación cruzada, las muestras de test hasta cinco veces, tal y como se dice en el párrafo anterior.

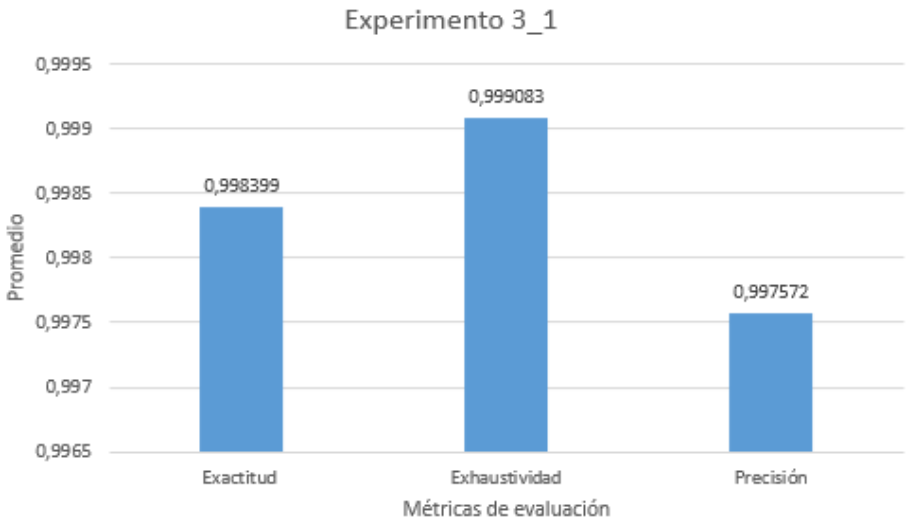


Figura 5.14: Primer ensayo del tercer experimento.

En el segundo ensayo se emplea el mismo método que en el anterior, pero intercambiando la distribución de las muestras de los dataset A y B, de manera que ahora es el conjunto de muestras de A el que se utiliza entero en el entrenamiento, mientras que de B se usa el 80%.

El 20% restante de muestras de B se destinan al test. Y al igual que antes, el conjunto de test se baraja, para que se escoja aleatoriamente hasta en 5 ocasiones.

Al observar los resultados en la Figura 5.15 se puede ver que se siguen obteniendo resultados realmente altos, mayores de 0.997. En este caso la exhaustividad alcanza el valor 1.0 debido a que no se obtuvieron falsos negativos en las diferentes evaluaciones.

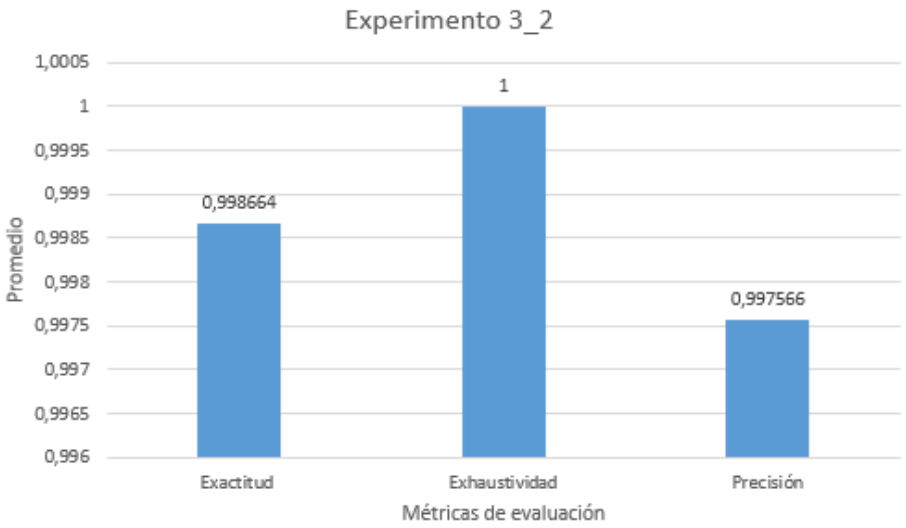


Figura 5.15: Segundo ensayo del tercer experimento.

En el tercer ensayo, se utilizan muestras de los tres sensores A, B y C. En concreto, en este caso se utilizan todas las muestras de A y B, es decir el 100% para entrenar. Mientras que se reservan todas las muestras del sensor C para la fase de test. En este ensayo y debido a que no se utilizan las mismas muestras para entrenar y para evaluar (son muestras de distintos sensores), no era necesario realizar validación cruzada. No obstante, las muestras de C también se han dividido en 5 grupos, se ha tomado al igual que en los ensayos anteriores el 20%, y se ha repetido el test, hasta en 5 ocasiones.

En este ensayo experimental, lo que se busca es observar la capacidad del sistema desarrollado de generalizar la detección táctil, entre muestras de diferentes sensores. Esto puede ser de interés, por ejemplo en situaciones en las que en un sensor se estropea o rompe el elastómero y requiere ser sustituido por otra unidad. Si la generalización es buena, permitiría evitar reentrenar el sistema.

En la Tabla 5.1 se pueden ver los resultados de cada una de las 5 iteraciones para el test con C.

	K=1	K=2	K=3	K=4	K=5	Promedio
Exactitud	0.504162	0.492271	0.529132	0.472057	0.505351	0.500595
Exhaustividad	0.980198	0.982673	0.985612	0.981333	0.992593	0.984482
Precisión	0.491925	0.485924	0.513109	0.457143	0.493252	0.488271

Tabla 5.1: Resultados del tercer ensayo.

Se puede observar en la Figura 5.16 una representación del promedio de los resultados de manera más gráfica.

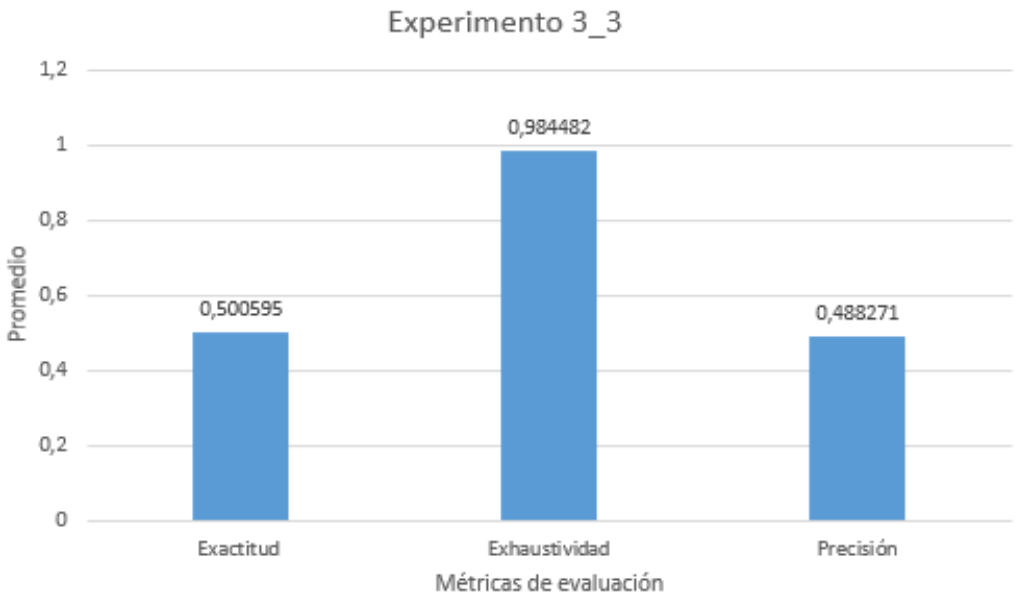


Figura 5.16: Tercer ensayo del tercer experimento.

Con estos datos se puede ver que los resultados decaen en promedio, obteniendo, por ejemplo, un resultado de 0.5 en el caso de la exactitud (accuracy). Si se comparan la exhaustividad (recall) con la precisión se puede observar que existe un mayor número de falsos positivos en comparación con el número de falsos negativos. Esto se corrobora observando la matriz de confusión en la que los datos muestran que se detectan más positivos, tanto verdaderos como falsos, que negativos, interpretando un mayor de datos como contacto. Se pueden ver

los datos de las matrices de confusión obtenidas en el experimento en la Tabla 5.2.

	K=1	K=2	K=3	K=4	K=5
TP	396	397	411	368	402
TN	28	17	34	29	23
TP	409	420	390	437	413
FN	8	7	6	7	3

Tabla 5.2: Resultados de la matriz de confusión.

En resumen, se puede contemplar que al generalizar los datos utilizando diferentes sensores para el entrenamiento y el test, los resultados disminuyen hasta encontrarse alrededor del 50% de acierto. Esto puede deberse a que las unidades de los sensores son construidas y ensambladas diferentes y los geles (elastómeros) se fabrican artesanalmente, haciendo que cada sensor tome imágenes de diferente manera. Si se pudiesen fabricar los sensores de manera industrial, probablemente se podría aumentar este porcentaje de acierto al crear sensores más similares entre sí. Otra opción, sería construir más sensores y aumentar el dataset para cada uno, o emplear un sistema o red generativa antagónica (Generative Adversarial Network (GAN)) que permitiera generar imágenes virtuales para distintas características de unidades DIGIT, y así entrenar el sistema de detección con los resultados de la GAN.

Miembros del grupo de investigación AUROVA realizaron un experimento utilizando estos datasets obtenidos con los sensores A, B y C. Este experimento empleaba, para la detección, una red convolucional profunda con la estructura base de una Inceptionv3.

Con esta red se obtuvieron resultados de un 97,6% en exactitud entrenando con los datasets A y B y evaluando el modelo con A. Además, se obtuvo un 99,35% en exactitud al entrenar de igual manera con A y B, pero testeando con B. Por otro lado, al testear con un sensor no utilizado en el entrenamiento los resultados obtenidos se encontraban entorno al 52% (Castaño-Amorós y cols., 2021).

Se puede observar que los resultados del experimento llevado a cabo en este proyecto y los obtenidos por AUROVA son bastante similares aun utilizando modelos de aprendizaje distintos.

- Cuarto experimento

En el cuarto experimento también existen tres ensayos. En ellos, se utilizan las muestras de los datasets de los sensores A, B y C para el entrenamiento. En cada ensayo se divide uno de los datasets en cinco grupos, se entrena con cuatro de ellos y se realiza el test con el restante, de manera análoga a las dos primeras partes del tercer experimento. De igual manera que en el experimento anterior, también se utilizó el algoritmo RF con 100 árboles para la realización de los ensayos.

Con esta prueba, se pretende observar los resultados al aumentar el número de muestras en el entrenamiento y ver cómo varían los resultados, si mejoran al aumentar el número de muestras que mejoren el modelo.

En el primer ensayo se divide el dataset del sensor A en cinco grupos y se procede como se ha explicado anteriormente, entrenando con cuatro grupos de A y los conjuntos B y C, mientras se evalúa con el grupo restante de A. Los resultados son bastante buenos, obteniendo un promedio en las tres métricas mayor del 0.998. Se pueden ver una comparación en forma de gráfica en la Figura 5.17.

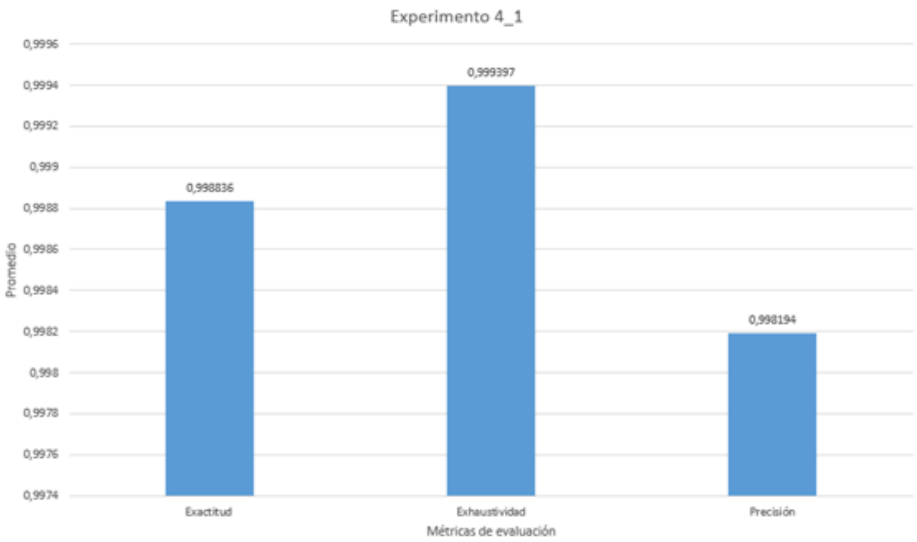


Figura 5.17: Primer ensayo del cuarto experimento.

En el segundo ensayo, B se divide en cinco grupos para usar cuatro para entrenamiento y el restante en test. A su vez, en el entrenamiento se introducen los datasets correspondientes

a los sensores A y C. Se obtienen valores aún mayores como se puede ver en la Figura 5.18, con valores por encima del 0.999 de promedio.

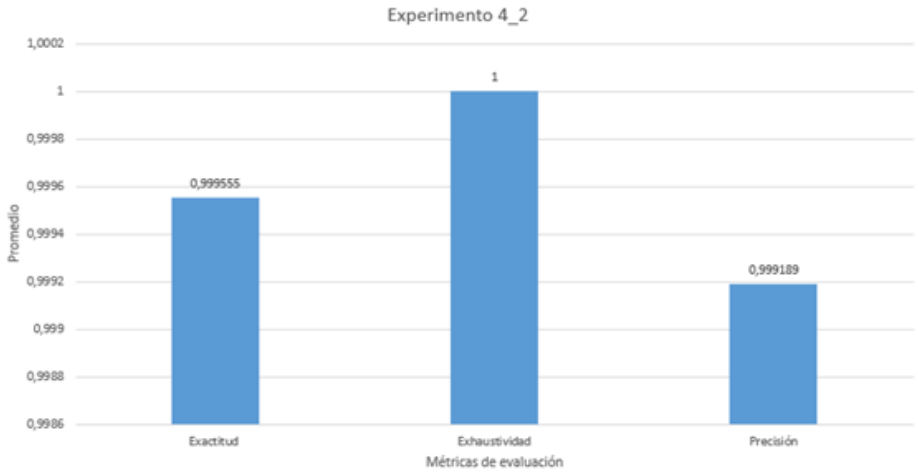


Figura 5.18: Segundo ensayo del cuarto experimento.

En el tercer ensayo, siguiendo el procedimiento, se utiliza un 20% del dataset C para test y el 80%, junto con B y A se utilizan para el entrenamiento. Se puede observar en la Figura 5.19, que el valor decrece hasta valores mayores de 0.984. Aún de esta forma, siguen siendo resultados realmente buenos.

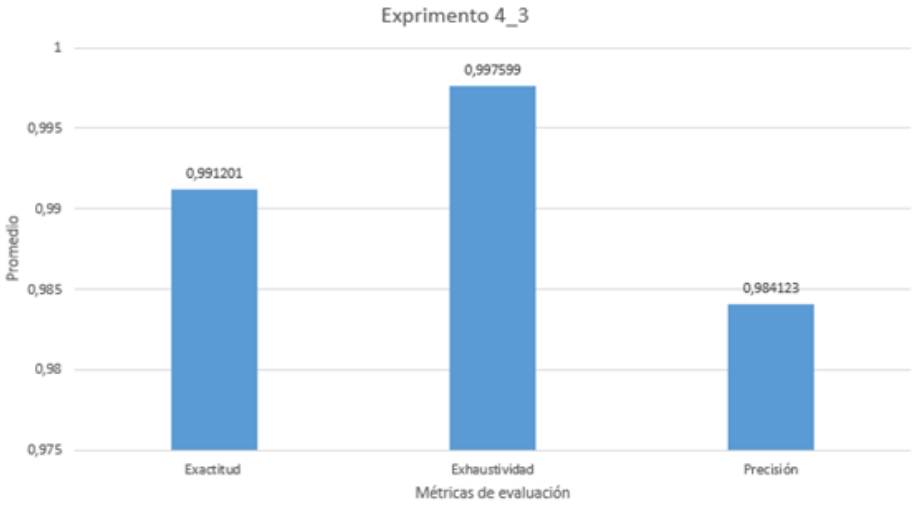


Figura 5.19: Tercer ensayo del cuarto experimento.

Se puede observar que los resultados disminuyen muy levemente, aunque esto es comprensible teniendo en cuenta que el número de muestras evaluadas en los anteriores experimentos es menor que en este último. Igualmente son resultados muy cercanos a 1, por lo que prácticamente se ha obtenido un número muy alto de aciertos, disminuyendo quizás más, en la evaluación de las muestras tomadas con el sensor C, que pueden resultar más diferentes dependiendo del sensor y de ser el primero que se creó.

6 Conclusiones y trabajo futuro

En este proyecto se ha dado a conocer el sensor táctil-óptico DIGIT. Para poder analizar su capacidad de sensorización, se necesitó, primeramente, realizar el montaje de sus componentes y fabricar algunos de ellos.

De manera análoga a la construcciones de los sensores se propuso el montaje del sensor en una pinza de la empresa Robotiq, por lo que se aprendió a moverla tanto de manera simulada, como de forma física con la pinza del laboratorio. Respecto al sensor se aprendió a tomar imágenes con él y, posteriormente, se analizarían diferentes datasets con imágenes tomadas en la Universidad de Alicante. Gracias a estas imágenes se pudo analizar el uso y fiabilidad del sensor. Debido a la novedad del sensor, una de las tareas principales era analizar su funcionamiento y observar la confianza que aporta a la hora de poder realizar agarre de distintos objetos.

Para obtener los resultados se propuso utilizar algoritmos de aprendizaje automático y más concretamente, de aprendizaje supervisado. Para ello, se observó la respuesta a los modelos creados de tres de los algoritmos más conocidos de esta área: KNN, RF y SVM. A partir de las imágenes del sensor DIGIT, se buscó el algoritmo que predijese mejor los resultados, aunque los tres eran realmente competentes.

En términos generales, se obtuvieron tasas de acierto muy elevadas, por encima del 98%. En el punto en el que más falla el sensor es el momento de generalizar, donde se utiliza un sensor distinto para las imágenes de entrenamiento y otro para las que se quiere obtener una predicción. En este caso se obtuvieron resultados de acierto entorno a un 50% aproximadamente. Esto se puede deber a la diferencia entre sensores, al tener que ser creados manualmente y no en un entorno industrial, donde serían iguales unos a otros.

Se puede concluir que se alcanzaron las metas propuestas durante la etapa inicial del proyecto de manera que se puede deducir que, si se utiliza un buen método de entrenamiento

del modelo a utilizar, DIGIT puede convertirse en una opción muy interesante y con un precio muy bajo en comparación a otros sensores del mercado, pudiéndolo poner en el punto de mira de muchas empresas e investigaciones relacionadas con este campo.

Los trabajos llevados a cabo en ese proyecto se han visto reflejados en mi participación como coautora en una publicación científica en las XLII Jornadas de Automática que se celebrarán en Castellón en Septiembre de 2021.

En un futuro, se podría ampliar esta investigación, comparando los datos obtenidos por los diferentes equipos de investigación de las distintas universidades que están realizando experimentos con el sensor, y más concretamente con el grupo de investigación AUROVA, de la Universidad de Alicante que me ha permitido realizar este proyecto de investigación dentro del proyecto europeo COMMANDIA. La comparación de los resultados puede resultar muy enriquecedora para las diferentes investigaciones.

Además, otra línea abierta de trabajo es integrarlo como parte de un controlador táctil de agarre más robusto, que además de emplear el sistema de detección de contacto desarrollado, incorporase nuevas funcionalidades como la detección de deslizamientos (resbalamiento de objeto entre los dedos de una pinza robótica).

Bibliografía

Anaconda. (2020). `https://www.anaconda.com/`.

Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning*, *3*, 19–48. doi: 10.5772/9385

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32. doi: 10.1023/A:1010933404324

Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., … Dollar, A. M. (2017). Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, *36*(3), 261–268. doi: 10.1177/0278364917700714

Castaño-Amorós, J., Gil, P., Fernández, I., y Puente, S. (2021). Detección de agarre de objetos desconocidos con sensor visual-táctil. *XLII Jornadas de Automática, Castellón 1-3 Septiembre.*

Coleman, D., Sucan, I., Chitta, S., y Correll, N. (2014). Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785.*

Fishel, J. A., y Loeb, G. E. (2012). Sensing tactile microvibrations with the biotac—comparison with human sensitivity. En *2012 4th ieee ras & embs international conference on biomedical robotics and biomechatronics (biorob)* (pp. 1122–1127). doi: 10.1109/BioRob.2012.6290741

Hu, H., Han, Y., Song, A., Chen, S., Wang, C., y Wang, Z. (2014). A finger-shaped tactile sensor for fabric surfaces evaluation by 2-dimensional active sliding touch. *Sensors*, *14*(3), 4899–4913. doi: 10.3390/s140304899

Kim, K. I., Jung, K., Park, S. H., y Kim, H. J. (2002). Support vector machines for texture classification. *IEEE transactions on pattern analysis and machine intelligence*, *24*(11), 1542–1550. doi: 10.1109/TPAMI.2002.1046177

Klatzky, R. L., Lederman, S. J., y Metzger, V. A. (1985). Identifying objects by touch: An "expert system". *Perception & psychophysics*, *37*(4), 299–302. doi: 10.3758/BF03211351

Kober, J., Bagnell, J. A., y Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274. doi: 10.1177/0278364913495721

Kolesar, E. S., y Dyson, C. S. (1995). Object imaging with a piezoelectric robotic tactile sensor. *Journal of microelectromechanical systems*, *4*(2), 87–96. doi: 10.1109/84.388117

Laaksonen, J., y Oja, E. (1996). Classification with learning k-nearest neighbors. En *Proceedings of international conference on neural networks (icnn'96)* (Vol. 3, pp. 1480–1483). doi: 10.1109/ICNN.1996.549118

Lambeta, M., Chou, P.-W., Tian, S., Yang, B., Maloon, B., Most, V. R., … others (2020). Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, *5*(3), 3838–3845. doi: 10.1109/LRA.2020.2977257

Learned-Miller, E. G. (2014). Introduction to supervised learning. *I: Department of Computer Science, University of Massachusetts*.

Lee, T.-W., y Lewicki, M. S. (2002). Unsupervised image classification, segmentation, and enhancement using ica mixture models. *IEEE Transactions on Image Processing*, *11*(3), 270–279. doi: 10.1109/83.988960

Li, J., Dong, S., y Adelson, E. (2018). Slip detection with combined tactile and visual information. En *2018 ieee international conference on robotics and automation (icra)* (pp. 7772–7777). doi: 10.1109/ICRA.2018.8460495

Li, S., Kwok, J. T., Zhu, H., y Wang, Y. (2003). Texture classification using the support vector machines. *Pattern recognition*, *36*(12), 2883–2893. doi: https://doi.org/10.1016/S0031-3203(03)00219-X

Maiolino, P., Maggiali, M., Cannata, G., Metta, G., y Natale, L. (2013). A flexible and robust large scale capacitive tactile system for robots. *IEEE Sensors Journal*, *13*(10), 3910–3917. doi: 10.1109/JSEN.2013.2258149

Mallapragada, P. K., Jin, R., Jain, A. K., y Liu, Y. (2008). Semiboost: Boosting for semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, *31*(11), 2000–2014. doi: 10.1109/TPAMI.2008.235

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, *24*(12), 1565–1567. doi: 10.1038/nbt1206-1565

Opencv. (2021). `https://opencv.org/`.

Oracle vm virtualbox. (2020). `https://www.virtualbox.org/`.

Pedreño Molina, J. L., Guerrero González, A., López Coronado, J., y cols. (2000). Estudio de los sensores táctiles artificiales aplicados a la robótica de agarre.

Programa Interreg Sudoe, F. E. d. D. R. F. (2020). *Commandia.* `http://commandia.unizar.es/`.

Robotiq. (2020). `https://robotiq.com/es/`.

Ros. (2020). `https://www.ros.org/`.

Schöpfer, M., Schürmann, C., Pardowitz, M., y Ritter, H. (2010). Using a piezo-resistive tactile sensor for detection of incipient slippage. En *Isr 2010 (41st international symposium on robotics) and robotik 2010 (6th german conference on robotics)* (pp. 1–7).

Scikit-learn. (2021). `https://scikit-learn.org/`.

Sferrazza, C., y D'Andrea, R. (2019). Design, motivation and evaluation of a full-resolution optical tactile sensor. *Sensors*, *19*(4), 928. doi: 10.3390/s19040928

Shimonomura, K. (2019). Tactile image sensors employing camera: A review. *Sensors*, *19*(18), 3933. doi: 10.3390/s19183933

Velasco-Sánchez, E., Zapata-Impata, B. S., Gil, P., y Torres, F. (2020). Clasificación de objetos usando percepción bimodal de palpación única en acciones de agarre robótico. doi: 10.4995/riai.2019.10923

Ward-Cherrier, B., Pestell, N., Cramphorn, L., Winstone, B., Giannaccini, M. E., Rossiter, J., y Lepora, N. F. (2018). The tactip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies. *Soft robotics*, *5*(2), 216–227. doi: 10.1089/soro.2017.0052

Lista de Acrónimos y Abreviaturas

ABS	Acrylonitrile Butadiene Styrene.
AUROVA	Automática, Robótica y Visión Artificial.
CAD	Computer-Aided Design.
COMMANDIA	Robótica móvil colaborativa de objetos deformables en aplicaciones industriales.
GAN	Generative Adversarial Network.
ICA	Individual Component Analysis.
KNN	K Nearest Neighbours.
LED	Light Emitting Diode.
PCA	Principal Component Analysis.
PCB	Printed Circuit Board.
PLA	Polylactic Acid.
RF	Random Forest.
ROS	Robotic Operating System.
SVM	Support Vector Machine.
TFG	Trabajo de Fin de Grado.